NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN OF USER FRIENDLY PROTOCOL TO EFFECT A TRANSPARENT INTERNETWORK TRANSACTION FACILITY THROUGH SPLICE AND THE DEFENSE DATA NETWORK

by

Stephen M. Carr

September 1985

Thesis Advisor: N. F. Schneidewind

Approved for public release; distribution is unlimited

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: Design of User Friendly Protocol to Effect a Transparent Internetwork Transaction Facility through SPLICE and the Defense Data Network	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis September 1985
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Stephen M. Carr	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School Monterey, California 93943-5100	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School Monterey, California 93943-5100	12. REPORT DATE: September 1985
	13. NUMBER OF PAGES: 144
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

transparent, internetwork, connectivity, requirements, SPLICE (Stock Point Logistics Integrated Communications Environment), DDN (Defense Data Network), DDS (Directory/Dictionary System), distributed, processing, protocols, TCP/IP, TCP (Transmission Control Protocol), IP (Internet Protocol), (Continued)

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This thesis explores transparent internetwork connectivity requirements for SPLICE (Stock Point Logistics Integrated Communications Environment). SPLICE nodes shall be interconnected via the DDN (Defense Data Network) to form a wide area network supporting distributed processing. Implementation problems, short and long term requirements for user terminal to serve host transparent connectivity, electronic mail, and process to process internetworking connectivity are presented. (Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S N 0102-LF-014-6601

Block # 19 (Continued)

MILSTRIP, user, friendly, transaction, facility

ABSTRACT (Continued)

Distributed processing through implementation of a DDS (Directory/ Dictionary System) is explored. MILSTRIP requisition referral by Stock Points through distributed processing is presented. Implementation of Tandem Corporation's EXPAND/X.25 protocols versus DDN TCP/IP and higher level DDN protocols are discussed. Packet switching, the CCITT X.25 and X.75 protocol standards, and an overview of the DDN TCP/IP protocols are offered in the Appendices.

Approved for public release; distribution is unlimited.

Design of User Friendly Protocol to Effect a Transparent Internetwork Transaction Facility through SPLICE and the Defense Data Network

by

Stephen M. Carr
Lieutenant Commander, Supply Corps, United States Navy
B. S., University of Minnesota, 1975

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

ABSTRACT

This thesis explores transparent internetwork connectivity requirements for SPLICE (Stock Point Logistics Integrated Communications Environment). SPLICE nodes shall be interconnected via the DDN (Defense Data Network) to form a wide area network supporting distributed processing. Implementation problems, short and long-term requirements for user terminal to server host transparent connectivity, electronic mail, and process to process internetworking connectivity are presented. Distributed processing through implementation of a DDS (Directory/Dictionary System) is explored. MILSTRIP requisition referral by Stock Points through distributed processing is presented. Implementation of Tandem Corporation's EXPAND/X.25 protocols versus DDN TCP/IP and higher level DDN protocols are discussed. Packet switching, the CCITT X.25 and X.75 protocol standards, and an overview of the DDN TCP/IP protocols are offered in the Appendices.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. SPLICE OVERVIEW

The SPLICE (Stock Point Logistics Integrated Communications Environment) concept is a revolutionary methodology for integrating an online and real time network of distributed computer systems into the Navy Supply System. To date, true online interactive processing between hosts at two geographically separated stock points has not been achieved. Currently, the few transactions that occur between two geographically separate hosts are executed in batch mode. The most pressing enhancement that SPLICE will bring to the Navy Supply System is mainframe processor relief at the NSCs (Naval Supply Centers) and NSDs (Naval Supply Depots) which together comprise the major stock points. The objectives for SPLICE can be found in the SPLICE Functional Description [Ref. 1: pp. 2-2, 2-3]. They will not be repeated here since they have been discussed at length in almost every other SPLICE related study.

B. STOCK POINT MAINFRAME SATURATION

Presently these Burroughs and Perkin-Elmer mainframes are saturated. It was felt that much could be done to alleviate the burden on these mainframes by installing front end processors to handle telecommunications processing and interactive queries from CRT (Cathode Ray Tube) terminals. These Burroughs and Perkin-Elmer machines are for the most part mid-size mainframes ranging from third generation to early fourth generation hardware. The hardware and operating systems employed in these machines are geared for efficient batch operation but not interactive processes.

Interactive processing has been retrofitted into the operation of these machines. However, the cost in CPU (Central Processing Unit) resources has been expensive and terribly taxing to the system, much to the detriment of overall operations. The present suite of hardware located at the major stock points is not able to properly support inventory control, contract management, financial accounting, requisition processing, transportation management and other automated logistic operations while concurrently handling interactive CRT processes.

C. SPLICE NEAR-TERM PAYOFF

The short-term payoff from the SPLICE hardware is not simply in the function of front end processing for telecommunications traffic. The expected payoff is in replicating frequently queried files and connecting these replicated files directly to the SPLICE hardware. It is estimated that ninety percent of all interactive CRT processes transacted against the stock point mainframes are simple queries. In this category of interactive processing, a question is asked of a file, but no records are changed, added or deleted. In other words, the file is not affected by the query, the users simply want to look at some pertinent records. Queries will be satisfied by duplicate files which are mirror images of the Burroughs or Perkin-Elmer files, on disk drives directly connected to SPLICE hardware. In this manner ninety percent of the CRT transactions can be handled by SPLICE hardware. This would obviate the need to access files on disk drives connected to the mainframes. Hence, a drastic reduction in CRT transactions executed on these mainframes should result.

D. SPLICE LONG-TERM PAYOFF

SPLICE long-term effectiveness and efficiency is outlined by the major categories that follow [Ref. 2: p. 2].

1. Telecommunications Support

SPLICE will establish a nucleus for supporting all present and future Navy logistic data communications requirements. Local data communications will be handled through a local area network. Long haul data communications will be provided through connection to the DDN (Defense Data Network). All of the SPLICE local area networks will be connected together, using the DDN as a backbone of one large virtual SPLICE network.

2. Interactive and Distributed Processing Support

The present mainframe hardware suite is not geared for interactive and distributed processing. The implementation of redundant front end multiprocessors for the current suite of mainframes will provide fault tolerance and graceful degradation of processing. These front end multiprocessors will handle all telecommunications processing and management of the local area network. They will act as the local gateway into the DDN for long haul data communications and manage the long haul process to process interfaces.

3. Economic Advantages of Hardware Standardization

Presently there exists a proliferation of various minicomputers that serve as front end processors for various functions to the Burroughs and Perkin-Elmer mainframes. These minicomputers are in varying stages of obsolescence and for the most part are incompatible with each other. Implementation of SPLICE multiprocessor hardware will

replace these obsolete and incompatible front end processors. This will reduce the cost of supporting multiple versions of hardware and software as well as reduce the personnel required for these operations.

E. RESEARCH OBJECTIVE

The major concerns of this research are the long-term payoffs identified in the previous section. Specifically this involves telecommunication support, interactive processing support and distributed processing support. Implementation of these objectives must be done in a way so that it is not obvious to the user that complex processes must be invoked to effect desired communications and transactions. This must be achieved in an intra-SPLICE environment, as well as with networks and processes residing outside the topology of SPLICE. A comprehensive data communications plan and concomitant protocols must be implemented to bring these objectives to fruition. This research will attempt to solidify the groundwork for this plan and its required protocols.

II. SPLICE DEVELOPMENT BACKGROUND

Several things have changed since the SPLICE Functional Description [Ref. 1] and System Specification [Ref. 3] were drafted. Some of these changes and their ramifications are discussed in this chapter.

A. DEFINITIONS

1. SPLICE Node

In this document, a "SPLICE node" refers to the array of Tandem NonStop TXP multiprocessors and the HYPERchannel local area network that comprise the backbone telecommunications hardware for each SPLICE installation. These hardware facilities are described below. The reason for reference to each SPLICE system hardware complex as a "node" is due to packet switched network parlance. A "node" roughly equates to a computer or system of computers located at one geographical location within the larger context of a long haul packet switched network. Strictly speaking, a "node" is a point of a network generally containing a switching element used to connect traffic, where various links come together [Ref. 4: p. 360].

2. Host

When referring to a software application process that is resident and executing on a CPU (Central Processing Unit), the machine running the application process is often referred to as a "host." Sometimes the words "host" and "node" are used interchangeably when referring to two machines at two geographically separate locations that are connected to each other over the long haul packet switched

network. A more formal definition of a "host" is a collection of hardware and software which is attached to a network and uses that network to provide interprocess communication and user services [Ref. 4: p. 352].

B. SPLICE CONTRACT AWARD

The SPLICE contract was awarded on 17 November 1983 [Ref. 5: p. 1]. It covers a time span of 15 years. The potential total amount of this contract is $548,380,876. The minimum implementation of this contract covers the installation of SPLICE hardware and software interfaces at thirty-five sites. This contract has the potential for acquisition and implementation of sixty-two SPLICE nodes in the event that funding and needs of the Navy warrant the additional installations [Ref. 6: p. 5-1].

1. Prime Contractor (Federal Data Corporation)

Federal Data Corporation is the prime contractor for SPLICE. They are responsible for the implementation and enforcement of standards for this large contract. They have subcontracted other corporations cited below to effect this.

2. Hardware Subcontractor (Tandem Corporation)

The Tandem Corporation is the prime subcontractor for SPLICE. Their main contribution is the SPLICE hardware, which is the Tandem NonStop TXP computer system. A summarization of salient features of this hardware system are cited in Tandem literature [Ref. 7: pp. 1, 2].

Each Tandem NonStop TXP system includes at least two processor modules, multiple controllers, multiple data paths between CPUs and input/output controllers, and multiple power supplies [Ref. 7: p. 2-2]. There is built in redundancy in the system due to parallel controllers and

parallel channels which facilitate dual paths and parallel flow of all information and data between CPUs and peripheral devices. All secondary storage devices are also parallel and redundant. Data is stored on two devices creating a mirror image of each other. Thus any single point of hardware failure will not stop an application process. The parallel channeling between CPUs is accomplished on Tandem's high speed DYNABUS interprocessor bus. There is a master - slave relationship between each pair of processors in the Tandem NonStop TXP system. In actuality, this paired relationship is application dependent, not a physically designated setup. Any process that is resident and running on a system, say system 1, is automatically backed up by its slave paired system, say system 2. If any failure of the primary system is recognized by the backup slave system, the backup slave system takes over processing. Upon recognizing a hardware failure in system 1, the original slave processor, system 2, now becomes the master processor. System 2 will now attempt to designate another available functioning system to back it up as its slave processor. Let us assume that system 5 is up and running and is available to assume the duty as a slave processor for system 2. System 5 now becomes the slave processor. In this manner, if system 2 should go down, then system 5, the new slave, will be able to resume application processing without interruption. Thus for reasons of redundancy and backup, Tandem NonStop TXP systems are always installed in a configuration of from 2 to 16 machines. If more than 16 machines are required, multiple banks of 2 through 16 more machines can be hooked together via Tandem Corporation's FOX fiber optic local area network. A typical SPLICE configuration will consist of 6 or more CPUs.

3. Local Area Network Subcontractor (Network Systems Corporation)

The Network Systems Corporation is the subcontractor for the local area network and attendant interfaces. They market a local area network product called HYPERchannel which is an HSLN (High Speed Local Network) implemented through a baseband bus [Ref. 8: pp. 143, 146, 163]. One of the main features of HYPERchannel is that it can achieve data transfer rates of up to 50 megabits per second. However, the most important feature of HYPERchannel with respect to SPLICE are the NIUs (Network Interface Units) that are implemented in the HSLN. These NIUs allow the interconnection of normally incompatible peripheral devices. The NIU transforms the data rate and protocol of the subscriber device to that of the local transmission medium and vice versa [Ref. 8: p. 208]. Through the use of these NIUs, the HYPERchannel is able to connect a myriad of processors and peripherals, allowing them to interface with the Tandem NonStop TXP SPLICE hardware system.

C. DEFENSE DATA NETWORK

The DDN (Defense Data Network) is the mandated long haul data communications common switched network for the Department of Defense. A pressing need was seen for a data communications network that would ensure an adequate level of reliability and redundancy in wartime, as well as ensure security of communications. The Department of Defense also sought to minimize the cost of rapidly expanding demand for data communications. As a result of these requirements and the need to modernize, improve and consolidate smaller existing data networks within the Department of Defense, the concept of the DDN was born.

1. Historical Perspective

In 1969, DARPA (Department of Defense Advanced Research Projects Agency) initiated an R&D (Research and Development) program for a packet switched data communications network [Ref. 9]. This network was called the ARPANET. The original goal of the ARPANET was development of an experimental network whose purpose was to advance the state of the art in computer resource sharing. It was to develop a communications network and procedures that would allow dissimilar computers at different geographical locations to communicate with each other. Through this network, hardware, software and data resources could be shared conveniently and economically by a wide community of users. As the ARPANET matured and the initial R&D goals of the network were achieved, users with operational requirements began to proliferate. By 1975, the responsibility for the ARPANET was transferred from DARPA to DCA (Defense Communications Agency). This was done because of the size of the network, and the increase of operational vice experimental systems being implemented on the network.

2. Demise of AUTODIN II

The Department of Defense has had a message switched data communications network operational for quite some time. This system is called AUTODIN. It is used heavily for plain text military message traffic. It is also used quite heavily by the military logistics community. Currently, the bulk of all MILSTRIP (Military Standard Requisition and Issue Procedure) transactions are transmitted through AUTODIN. Several years ago, it was recognized that the AUTODIN system was getting severely overloaded and had to be expanded or replaced. The system designated to enhance and replace AUTODIN was called AUTODIN II. For several years an

effort was made in the direction of AUTODIN II implementation. However, DARPA's ARPANET project had proven to be so successful that strong arguments surfaced for implementation of a data communications system based on ARPANET's architecture.

The assumptions under which AUTODIN II was planned were no longer valid in view of the changing requirements and costs. The small number of nodes planned for AUTODIN II were not deemed capable of survival to a satisfactory degree in wartime. In addition, it was doubtful that the switches could be certified to handle traffic at all security levels. Finally AUTODIN II appeared to be too expensive. Also, the level of common carrier tariffs was increasing to the point as to make long access lines required for the limited number of nodes prohibitively costly.

The AUTODIN II project got bogged down. Finally, on 2 April 1982, the Deputy Secretary of Defense directed the AUTODIN II project terminated [Ref. 10], [Ref. 11: pp. iii, iv]. Furthermore, he instructed the Director of DCA to proceed immediately with the development of the Defense Data Network as outlined in the January 1982 ARPANET Replica Program Plan.

3. Merging of Existing Networks into the DDN

In 1983, DCA split the old ARPANET into two separate networks. As a result of the split, there is now an ARPANET for the research community and MILNET, an unclassified segment of the DDN, for the military community. In the near future, the MILNET, which is a spin off from the ARPANET, and the MINET (Movements Information Network), located in Europe, will be integrated into the MILNET to form the unclassified backbone segment of the DDN.

The classified segment of the DDN is to be built on the foundation of the Secret Network. The WINCS (WIN

Communications Subsystem), SACDIN (Strategic Air Command Digital Network), DODIIS (Department of Defense Intelligence Information System) and TS will all eventually be integrated into the backbone of the Secret Network.

4. Mandate for DDN as DoD's Long Haul Common Switched Data Network

The following directives mandate implementation and use of the DDN as the common switched data communications network for the Department of Defense. In accordance with these directives, interconnection of long haul SPLICE nodes is to be accomplished through the DDN. Only by showing that the DDN is incapable of supporting the needs of SPLICE can a case be made for implementing an alternative long haul data communications arrangement.

a. Deputy Secretary of Defense Memorandum of 2 April 1982

This document directed termination of AUTODIN II and initiated implementation of the DDN. The memorandum states [Ref. 10],

> "It remains DoD policy that all data communications users will be integrated into this common user network. Exceptions to this policy must continue to receive the approval of the Deputy Under Secretary of Defense for C3I."

b. Under Secretary of Defense for R&E Memorandum of 10 March 1983

This document reinforces the DDN mandate stating [Ref. 12],

> "Existing systems, systems being expanded and upgraded, and new systems or data networks will become DDN subscribers. All such systems must be registered in the

DDN User Requirements Data Base (URDB); Once registered in the URDB, requests by a service/agency for an exception to this policy shall be made to DUSD C3I."

c. OPNAV Instruction 2070.4 of 7 March 1984

This document provides guidelines for Department of the Navy implementation [Ref. 13].

". . . the DDN will be used by all DOD ADP systems and data networks requiring interconnection by telecommunications. Any requests for waivers from this policy must clearly show why DDN cannot meet the requirement. Waivers for periods of more than 2 years can only be approved by Deputy Under Secretary of Defense, DUSD (C3I). Waivers for periods of less than 2 years can only be granted by DCA if CNO and DCA agree that it is justified.

This instruction applies to all Navy ADP systems and data networks requiring data communications services. Long-haul and area communications, interconnectivity, and the capability for interoperability will be provided by the DDN. This includes existing ADP systems, ADP systems being expanded and upgraded and new ADP systems.

All commands will ensure future ADP acquisitions which require data communications include provisions for using the DDN as their primary data communications medium."

Promulgation of the three documents cited above, in particular the last one, provide the rationale for SPLICE long haul data communications through the DDN.

5. Growing Pains in the DDN

It can fairly be argued that the scope of service that DDN is ultimately to offer eclipses most other packet switched network implementations with the possible exception of the commercially available Tymnet and GTE Telenet. However, the DDN goes much further than these two commercial systems. Tymnet and GTE Telenet offer a method of point-to-point transmission. However, unlike DDN, they do not offer implementation of higher level protocols which will ensure reliable transmission and process-to-process

communication. In this respect, the DDN is a large capital venture that rides the leading edge of technology in a new applications area.

As of July 1985, there were approximately 90 switches in the DDN. Between July 1985 and April of 1986, another 182 switches are to be installed. Clearly, the DDN is in the midst of massive growth. With it come growing pains. The basic architecture and technology for implementation of the DDN has been proven through the ARPANET. However, the ARPANET implementation was relatively slow and gradual over a period of fifteen years. It is dwarfed in comparison with the scope and rapidity of DDN implementation. Therefore, it can be expected that actual physical installation and implementation problems will trouble DCA over the next couple of years. The three directives cited in the previous section illustrate the urgency with which the Department of Defense intends to convert existing and new data communications applications to the DDN. This will entail no small effort, and already implementation scheduling problems have emerged. SPLICE is no exception to this problem.

D. RECENT DEVELOPMENTS AND DDN IMPLEMENTATION COMPLICATIONS

As alluded to previously, implementation of SPLICE long haul data communications through the DDN is not a simple matter. Full implementation into the DDN requires the implemention of different hierarchical protocols which map out a standardized methodology for data communications. Detailed discussions of these protocols are presented in Appendix B. It should be noted that few hardware vendors actually have the DDN protocol software available for their suite of hardware. The majority of the DDN protocol implementations currently available stem from ARPANET which

uses the same protocols. The number of hardware specific DDN protocol implementations are growing, and DCA is awarding contracts to various vendors and software houses to develop these protocols for certain major brands of hardware. However, Tandem Corporation is not one of the vendors that could offer these DDN protocols off the shelf to run on their Tandem NonStop TXP system hardware. Consequently, the SPLICE contract had to be written to include development and implementation of these protocols [Ref. 5: pp. 58, 186], [Ref. 14: p. 6]. Other implementation problems are addressed below.

1. TSR (Telecommunications Service Request) Backlog

TSRs are the vehicle by which telecommunications lines are ordered by the armed services and agencies of the Department of Defense. TSRs are centrally coordinated and handled for the Department of Defense by the Defense Communications Agency. The rapid increase in distributed computer systems applications, local area network implementations and connectivity requirements into the DDN have resulted in a logjam of TSRs forwarded to DCA. The sheer increase in volume accounts for some of the delay in getting connections for ADP applications. Moreover, divestiture of AT&T has created another problem as cited by the 15 May 1985 DDN Newsletter [Ref. 15: pp. 1, 6].

> "Circuit acquisition (both access and trunk circuits) continues to be a problem. The regional BOCs (Bell Operating Companies) are reluctant to do business with ATTIS (American Telephone & Telegraph Information Systems); consequently we have over 230 circuits backlogged with no clear idea of when this log jam will be cleared. The Commercial Policy folks at DCA as well as the DCA Regulatory Counsel are working on this issue.
> . . .
>
> In FY 1983, a total of 521 DDN TSRs were processed, and in FY 1984, 1290 DDN TSRs were issued. So far in FY 1985, a total of 800 DDN TSRs (Oct-Jan) were processed, with an estimation of 3,000 DDN TSRs to be processed during the FY 1985 year. Presently, this section processes approximately 40 DDN TSRs to the field each

week, including user requirements, backbone trunks and dial up service lines to the TACs (Terminal Access Controllers)."

NAVSUP (Naval Supply Systems Command) Headquarters has a DDN waiver for SPLICE that expired on 31 March 1985 [Ref. 16], [Ref. 17]. Due to a myriad of scheduling problems, not the least of which is the difficulty in processing TSRs and obtaining connections into the DDN, an extension of this waiver has been requested through COMNAVTELCOM (Commander, Naval Telecommunications Command) [Ref. 18], [Ref. 19]. Because the waiver has expired, NAVSUP currently has no legal vehicle by which to obtain telecommunications connections for SPLICE installations. A dozen SPLICE to DDN connection TSRs were backlogged in July 1985. The average age of these TSRs is about 9 to 12 months; some of them are as old as 15 months. In early July 1985, the first two SPLICE TSRs were finally completed. These were for a 56,000 bit/second connection from the FMSO (Navy Fleet Material Support Office) Mechanicsburg, Pennsylvania SPLICE site number 1 into the DDN, and another 56,000 bit/second connection from NARDAC (Navy Regional Data Automation Center) Jacksonville, Florida SPLICE site into the DDN. Unfortunately, DCA was not able to physically deliver a 56,000 bit/second line into FMSO, so a 9,600 bit/second line was substituted until a 56,000 bit/second trunk could be negotiated and connected. Thus at the time of writing this document, there are only two SPLICE DDN connections physically in existence.

A self contradicting situation has ensued. Federal Data Corporation as the SPLICE prime contractor is responsible for the development and implementation of the DDN protocols cited earlier. To implement and test these protocols, the contractors' hardware test beds must be connected into the DDN. Two TSRs requesting connection of

Federal Data Corporation (TSR DU10MAY840576) and Tandem Corporation hardware test beds (TSR DU07AUG841132) have been backlogged since 2 May 1984. As a result, the prime contractor does not have the physical connections necessary to implement and test the DDN protocols currently under software development. Until these protocols are tested and implemented, SPLICE is unable to break the yoke of having to operate under a DDN waiver. At the time of writing, the request for DDN waiver extension [Ref. 18] was still unresolved. Yet, even if the SPLICE DDN protocols were up and running, SPLICE would be unable to function as a wide area network because of the physical unavailability of DDN connections due to backlogged TSRs. Neither can SPLICE operate outside the DDN as a wide area network using dedicated lines leased from commercial carriers, because the DDN waiver has expired.

2. DDN Asynchronous Terminal Data Communications Support

As shall be noted later in the discussion of protocols, the DDN is configured to support only asynchronous terminal data communications. Asynchronous communication has tremendous advantages due to simplicity of operation and implementation. In asynchronous mode, characters are transmitted one at a time, and the rate that the data is transmitted is variable. This means of communication is simple and cheap, but requires an overhead of 2 to 3 bits per character [Ref. 8: p. 26]. The overhead required is for start and stop bits. Normally a constant bit stream of 1s is sent over the line. When the receiver notes a transition from 1 to 0, this indicates the beginning of a character. This 0 bit is called a start bit. After the character is transmitted, the sender resumes sending 1 bits down the line until the next character is to be

transmitted. This translates to a relatively high rate of overhead, a minimum of 25% to 37.5%, for 8 bit byte character transmission. Unfortunately for SPLICE, very few of the presently installed CRT terminals at the major stock points are asynchronous.

3. Burroughs Bisynchronous Terminal Data Communications Support

The major stock points have CRT terminals installed that are connected to the Burroughs mainframe. These CRT terminals run on Burroughs bisynchronous transmission (poll/select) protocol. As one might infer from the name, bisynchronous transmission requires accurate clocking so that the transmitter and receiver are in step, or synchronized. In this method of transmission, blocks of characters are transmitted without the start and stop bit codes encountered in asynchronous transmission. The exact departure and arrival time of characters is known. The beginning and end of each block of characters transmitted is delimited by synchronization characters. These synchronization characters are entirely different in type from normal data that is to be transmitted [Ref. 8: p. 27].

There are two big advantages to this type of terminal data communications. First of all, it is possible to communicate in a screen mode. That is, the entire screen of a CRT display is used, enabling displays of menus and format templates which are essential for proper editing of data entered by clerical personnel. The templates may show the proper format for data entry, and guide the clerical personnel from data entry block to data entry block. A full screen of data on a normal 24 line, 80 character per line terminal is 1920 bytes of information. The second big advantage to synchronous communication is the high rate of data transfer that can be attained. Since it is a clocked

system, and blocks of data can be transmitted, the entire 1920 bytes of screen data can be transmitted in one block.

One of the big problems that the DDN currently poses for SPLICE is the inability to support the Burroughs bisynchronous terminal communications. What this means is that a Burroughs terminal connected to a SPLICE node at say, NSC Oakland, CA, will be unable to communicate via the DDN with the Burroughs mainframe at NSC Norfolk, VA. Moreover, communications between an asynchronous terminal connected into the DDN through a TAC (Terminal Access Controller) would be unable to converse with a Burroughs host at a SPLICE node using the Burroughs bisynchronous data communications protocol.

This obstacle can be overcome, but it will require development of telecommunications software to wrap this sort of communication up and carry it between Oakland and Norfolk. If a smart terminal is connected to a SPLICE host via a DDN TAC, this same sort of software would be required to wrap and unwrap the Burroughs bisynchronous terminal data communications protocol. It is not however, feasible to invoke this bisynchronous protocol between a SPLICE host and a dumb terminal. At this time, no such software is developed, and there are no plans to develop it. The Burroughs hardware is scheduled to be replaced as a result of the SPAR (Stock Point ADP Resolicitation) project. Investment of software development into a system that is likely to be replaced in 3 to 5 years is not being considered. Furthermore, the bisynchronous method of terminal data communications is rapidly becoming obsolete and is being replaced by a synchronous data link protocol, which is discussed below.

4. IBM SNA SDLC Synchronous Terminal Data Communications Support

One of the items that is never mentioned in the SPLICE Functional Description [Ref. 1] or the SPLICE Systems Specification [Ref. 3] is the requirement for SPLICE to act as a front end processor for the IBM 3081 series mainframes that are being installed at the ICPs (Inventory Control Points). These IBM 3081 machines communicate with IBM terminals in a local area network using IBM's SNA (Systems Network Architecture) SDLC (Synchronous Data Link Control) protocol. Closely coupled with this requirement is the need for the ICPs to communicate with the Trident Refit Facilities and to interface with the CAIMS (Conventional Ammunition Integrated Management System). These applications are apparently also going to utilize IBM 308X series hardware. So the scope of communications problems no longer is limited to applications currently being supported by the Burroughs and Perkin-Elmer mainframes at Stock Points. Now we have a communications problem between different communities outside the purview of SPLICE. Again, the problems that synchronous terminal data communications pose for SPLICE are similar to those cited previously for the Burroughs bisynchronous terminal data communications problem.

The IBM SNA SDLC is a bit oriented synchronous mode of communications that is rendering the character block bisynchronous mode obsolete. This bit oriented scheme treats the block of data to be transmitted as a continuous bit stream rather than a character stream. The synchronization characters for delimiting the beginning and end of block transmission cannot be used, since all transmitted data is assumed to be an arbitrary bit pattern, and such patterns may be legal data to be transmitted. SDLC

uses the bit pattern 01111110 as a preamble and postamble delimiting bit pattern to accomplish the same task that the synchronization characters provided in bisynchronous communications. The problem of 6 consecutive 1 bits appearing in a data stream as data and being misinterpreted by the receiver as a preamble or postamble bit pattern must be solved [Ref. 8: p. 27]. It is solved by a procedure known as bit stuffing. The transmitter will always insert an extra 0 bit after any occurance of five 1s in the data being transmitted. In the event that the pattern 01111110 (the left end being the start of the pattern) is actually sent as data, the transmitter will insert a 0 after the fifth 1 so that the data sent is 011111010. When the receiver gets the data it always examines any set of five consecutive 1 bits. When it detects five 1 bits, it checks the next bit to see if it is a 0. If it is a 0 bit, the receiver deletes it. In this manner, the only reason that 6 consecutive 1 bits will be transmitted is if it is either the preamble or postamble bit pattern delimiter.

5. Interface With European MILNET Terminals

The European section of the MILNET, still better known as the old MINET, also poses a possible communications problem for SPLICE. There is a gateway between the European MILNET (MINET) and the CONUS (Continental United States) portion of the MILNET. In order to process any traffic through this gateway, IP (Internet Protocol) and TCP (Transport Control Protocol) must be implemented. It is not possible to implement communication with users or hosts in this subsection of the DDN by using the CCITT (Consultative Committee for International Telephone and Telegraph) X.25 standard and a commercial vendor higher level protocol [Ref. 8: p. 41]. This negates the ability for SPLICE to implement an off the shelf set of network protocols

available from Tandem Corporation to communicate with nodes in the European MILNET.

III. IMPACT OF THE PROPOSED TP-4 STANDARD

One of the problems associated with the onset of distributed processing is the standardization of protocols. For the commercial environment, layers 1 through 3 (physical, data link, and network) have largely been standardized through the CCITT X.25 specification. As mentioned in Appendix A, X.75 offers internetwork connectivity of X.25 networks. In the ARPANET environment, standardization for layers 1 and 2 (physical and data link) are based upon the ARPANET HDLC (High Level Data Link Control) and the 1822 specification [Ref. 20]. In the ARPANET and DDN, the IP and TCP functions cannot be accurately mapped out as layer 3 (network) or layer 4 (transport). In fact, between these two protocols many of the functions defined for layers 3 through 5 (session) are covered, and the correspondence between the ISO OSI model and TCP/IP breaks down. As of yet, there is still no commercially implemented standard for layer 4 (transport). However, the DoD and the NBS (National Bureau of Standards) have been working toward development of protocol standardization. The NBS ICST (Institute for Computer Sciences and Technology) in cooperation with the DoD, the ISO (International Standards Organization), and many industrial firms, have developed new international standards. Two new protocols that are standardized are a new Internetwork Protocol and the TP-4 (Transport Protocol) [Ref. 21]. The U. S. standards organizations are supporting TP-4 in international operations, and the Department of Commerce is proposing that TP-4 become a FIPS (Federal Information Processing Standard) for the Federal Government.

Although TP-4 is based upon TCP, there are significant dissimilarities that would cause conversion to be a complicated matter if DoD were to mandate TP-4 use in existing TCP applications. TCP was developed and first promulgated in 1978 and thus has a long and proven operational history. TP-4 has yet to be implemented. It is expected that a commercial TP-4 product will be available in the near future. TCP/IP were specifically developed to support DoD data communication demands in a hostile wartime environment. To this end, TCP/IP support the following requirements.

- Survivability: Some networks must function, even if at reduced performance, after many nodes and links have been destroyed.
- Security: Traffic patterns and data must be selectively protected through encryption, access control, auditing and routing.
- Precedence: Systems should adjust the quality of service dependent upon the basis of priority; Capability for preemption by higher priority must be available.
- Robustness: The system must not fail or suffer much loss of capability due to unpredicted situations, unexpected loads, or misuse.
- Availability: Elements of the system needed for operational readiness or fighting must be continuously available.
- Interoperability: Different elements of the DoD must be able to communicate with one another, often in unpredictable ways, between parties that had not planned to interoperate.

The operational needs cited above translate to five technical and managerial needs for the DoD.

- Functional and operational specifications.
- Maximum interoperability.
- Minimum procurement, development and support costs.
- Ease of transition to new protocols.
- Responsiveness to changing DoD requirements.

TCP and TP-4 are functionally equivalent, providing essentially similar service, though their architectures are dissimilar. Although there are differences between the two

protocols, the NRC is confident that TP-4 will meet military requirements. The DoD could incur significant long run savings through implementation of an international standard with a wide commercial application. On the other hand, a significant investment in TCP has already been made, and transition of current TCP applications to a new TP-4 standard could prove prohibitively costly. However, the DoD has a need for functional interoperability with a wide number of agencies and activities external to it. These agencies and activities are committed to the ISO standards. They include NATO (North Atlantic Treaty Organization), different intelligence and security agencies, and other segments of the Federal Government. The National Research Council, whose report is discussed herein [Ref. 21: p. 8], states, "The same objectives that have prompted the use of standardized protocols at higher level headquarters will lead to their use by tactical groups in the field."

"The Internet Protocol part of the standards is not believed to be a problem. The ISO IP is not as far along as TP-4, but it is much less complex." [Ref. 21: p. 9]. The progress being made in development of the ISO IP and TP-4 is very rapid. This is due in large part to a strong commercial demand for integration of data processing and networking, and the efforts of vendors to meet this new demand. In spite of the fact that the DoD was instrumental in development of TCP, which is a model upon which the ISO TP-4 is built, there is little chance that DoD will have much influence in altering TP-4 to conform with TCP. This is mainly because the DoD represents a small fraction of the total market for TP-4 implementation, and because the United States supports the ISO standard. Thus DoD was faced with making a decision regarding the implementation of TP-4 and the future of TCP.

The NRC offered three options to DoD, and recommended implementation of option 1 [Ref. 21: pp. 18 thru 21].

> "Option 1: The first option is for the DoD to immediately modify its current transport policy statement to specify TP-4 as a costandard along with TCP. . . .
>
> Option 2: Under option 2 the Department of Defense would immediately announce its intention to adopt TP-4 as a transport protocol costandard with TCP after a satisfactory demonstration of its survivability for use in military networks. A final commitment would be deferred until the demonstration has been evaluated and TP-4 is commercially available. . . .
>
> Option 3: Under the third option the DoD would continue using TCP as the accepted transport standard and defer any decision on the use of TP-4 indefinitely."

The Assistant Secretary of Defense for Command, Control, Communications and Intelligence rejected the NRC recommendation of Option 1, which would immediately mandate TP-4 as a costandard with TCP [Ref. 22], and ultimately move toward exclusive use of TP-4. Instead, DCA was directed to study and implement option 2. The meaning to all users of the DDN is that there is no current definitive strategy that they can count on five to ten years hence. At the moment the DoD TCP/IP is the standard but it is clear that this may change over the next decade. In any event, it is clear that DDN will support the DoD TCP/IP for at least the next decade. This affects SPLICE internetworking and higher level protocol implementation strategy. If software development of internetwork processes and remote online processes place a heavy dependency on the inner workings of TCP/IP, significant coupling between application modules and TCP/IP modules will result. This will make conversion to a different standard some 5 to 10 years hence costly and painful and will hamper any resolicitation and replacement efforts involving SPLICE.

IV. SHORT-TERM REQUIREMENTS FOR TRANSPARENT INTERNETWORK CONNECTIVITY THROUGH SPLICE

As stated at the beginning, the long-term payoff for SPLICE lies in its ability to support distributed computing. The long-term goal of SPLICE should be to effect transparent processing for the user. SPLICE should satisfy a user requirement without concern on his part with processes or machines to be used. The user should be dealing with the entire logistics system as an entity, and he should be able to interface directly with all facets of it in a global fashion. Software has not been developed yet to implement such internetwork process transparency. As of now more traditional specific machine and site dependent processes must continue. In the interim phase, there will be a need to access and conduct transactions against specific SPLICE nodes. This chapter will address the aspects of invoking such connections and transactional processes. Chapter V introduces the requirements for effecting true transparent interprocess connectivity, a long-term goal for SPLICE and a concept that will eliminate the need for many of the procedures to be discussed in this chapter.

A. USER TERMINAL TO REMOTE PROCESS CONNECTIVITY

This method of interprocess communication will most visibly change the way customers, expediters, and clerical personnel at the stock points and ICPs do business.

1. Stock Check of Material at Stock Points and ICPs

It has been the long-standing goal of supply personnel in direct support of repair and maintenance to be

able to query the records of local and remote stock points and ICPs in search of high priority repair parts. This is commonly referred to as performing a "stock check." It is the bread and butter of all supply personnel engaged in expediting repair parts. Even before the implementation of SPLICE, a select few terminals and internetworking systems have been implemented to provide limited query capability of the ICPs and stock points at major commands, most notably the Type Commands. Primarily however, this capability exists among a select few stock points and ICPs. For instance, SPCC has the ability to query stock levels at DLA (Defense Logistics Agency) Depots. Supply clerks engaged in direct support of an organizational maintenance effort currently do not have the ability to perform remote stock checks. SPLICE will change that.

2. Requisition Status Queries at Stock Points and ICPs

The second major query transaction of interest to customers in direct support of maintenance is "Requisition Status." Material has been ordered from the system, and the requisition is in process. This type of query is typically generated when a repair part is identified as being urgently needed for repair of equipment. The repair effort is in a "workstoppage" situation that is frustrated for lack of the part, and can proceed no further until it arrives. Queries of requisition status attempt to identify the precise state of nature that the requisition is in, which activity is currently handling it, and what is the expected delivery date of the material.

3. Administration of User Terminal Accounts and Passwords within SPLICE

For the most part, the applications requested will be stock checks and requisition status queries. These are

likely to be the only capabilities that the most rudimentary account holder, hereafter referred to as an "anonymous public domain account," would possess. Applications involving execution of transactions would require a higher level of account capability. These are application capabilities that clerical personnel at the ICPs, stock points, and major commands would require. The temptation may be great to implement a login procedure from a user host to a server host that will give blanket account authority to a certain class of terminals. This must be avoided and not implemented. A specific example follows.

Suppose a bank of 20 terminals at NSC Oakland are physically located in a specific office such as "Requirements Department." One might be tempted to conclude that no one without a legitimate need to conduct transactional applications against a remote stock point server host would ever be working on these terminals. It may be expedient from a software engineering standpoint to program a general login procedure from any of those 20 user terminals with a remote stock point server host. This login procedure would automatically link to an account at the server host with higher level application capabilities. What is the premise for this action? Presumably, when a terminal in "Requirements Department" is activated, a local login procedure is executed, and a bonafide "Requirements Department" clerk has gained access to the local SPLICE system with capabilities commensurate with his account. So why go through a complex login procedure with a remote server host once this is effected? Simply this. The person who originally logged into the local SPLICE host may have walked away from the terminal hours ago leaving the terminal unattended. Even if there is a timeout mechanism for automatically logging out an inactive terminal there will always be a window available for an illegal user to

penetrate the local system. If he succeeds in this, he may attempt to penetrate a remote stock point or an ICP. He will again succeed if the terminal he is illegally using will automatically login to an account with high level application capabilities at remote server hosts. Hence the need for the requirement that all accounts at local SPLICE systems as well as remote SPLICE systems have login procedures that are driven by individual accounts and passwords. The integrity of remote SPLICE node data will inevitably be compromised if a tight system wide login protocol is not enforced for all but anonymous public domain accounts.

Each terminal user will have his own account and password. It is recommended that a user's password be recognized and universally honored by all SPLICE nodes. This will facilitate easy access to the myriad of different high level accounts a user may possess at different SPLICE nodes. The user should be able to change his password whenever he feels it is necessary and still have it universally honored by all SPLICE sites. Such a password management process would necessarily require centralized accounts and a password management software module.

One possibility for account implementation would be for all high level accounts with remote capability to be centrally administered, say by FMSO. The accounts and passwords could be revalidated and repromulgated dynamically to all SPLICE nodes concerned. This process of updating the accounts and passwords on a real time basis may be somewhat involved, but it could be accomplished using end-to-end encryption offered by the DDN to update account and password tables at different SPLICE sites. If an account linked to a certain person is found to be abused, information could be transmitted from the central account administrator to disable that account almost immediately at all SPLICE nodes.

A drawback to such a scheme is that one might find a way to penetrate the accounting systems software, and effectively disable everyone's capabilities. Sufficient care must be exercised in design of such a system and its modules to minimize this threat.

A second method for implementing high level account control would be for each SPLICE site to administer and authorize high level accounts used at its site. Under such a system, the Commanding Officer would have more control as to which people external to his command have high level transaction application capabilities. The drawback of this system is that it would not have uniform criteria for granting accounts, and some systems would have easier access than others. In extreme cases, the management of high level accounts at a certain command might be so restrictive as to impede the normal flow of distributed processing.

The bottom line is that with the exception of anonymous public domain accounts, all user terminal accounts must be strictly linked to individuals. Implementation on a system wide basis is a problem that must be addressed and solved if the benefits of distributed processing are to accrue while concurrently maintaining adequate security.

One may argue that there is no instance whereby supply personnel at a remote terminal would ever have the need to execute high level application transactions against a geographically distant stock point. Therein lies the double edged sword of distributed processing. It allows greater system wide visibility and efficiency, while compromising the local control over files that heretofore were the exclusive domain of a stock point or ICP. It is inevitable that the Fleet Commanders will want higher level application capabilities, such as releasing critical material from war reserve stock. In a distributed environment, they will demand this capability and eventually

they will get it. Such capabilities will be demanded on down the line until at some level external to the supply system they are cut off. Perhaps it will be at the Type Commander level. A scenario can be envisioned whereby COMNAVLOGPAC will have certain universal transactional capabilities that would exceed the scope of those granted to the Force Supply Officer at COMNAVAIRPAC.

4. Requirements for Local SPLICE User Terminal Connection to a Remote SPLICE Server Host

In this situation, a terminal is directly hooked into a local SPLICE node. This may be a coaxial connection using Burroughs bisynchronous communications or IBM SNA SDLC. It may also be an asynchronous terminal hooked to a SPLICE node via the telephone PSN (Public Switched Network) and a modem connection. It is assumed that local login procedures to the local SPLICE node are well established, therefore they will not be addressed further. The problem begins when the local user terminal requires connectivity with a remote server host. Several things must happen.

- The user terminal indicates to the local SPLICE host that a connection is desired with a remote SPLICE host.
- The local SPLICE host, henceforth called the "user host," must establish a logical connection with the remote SPLICE host, henceforth called the "server host."
- Once the connection is established, a "login procedure" at the server host must be executed for the user terminal.
- The connection will either be honored or rejected by the server host dependent upon server host login criteria.
- If login is successful, the user terminal is now in a position to execute applications commensurate with the user's account capability.

Using the centralized account and password management criteria proposed earlier, the terminal user would have to enter his account data and password for transmission to the server host. However, the criteria for

accomplishing this connection should be limited to a request for connection to a specific SPLICE host and the entry of account data and a password. Account data should be simple. The user's name is recommended. Entry of the password should not print out or display at the terminal. This brings up the question as to how the user must specify the server SPLICE host to be accessed. There will be from 35 to 62 SPLICE nodes eventually. A "User Terminal To Server Host Transparent Connectivity Module," hereafter referred to simply as the "Connectivity Module," should be accessed. The Connectivity Module would offer selection menus, one of which would display a table of all of the different SPLICE nodes. The user should be able to browse the SPLICE table and select the server host that he wishes to connect with. The Connectivity Module would then effect the logical connection between the user terminal and the server host. After the logical connection is successfully effected, the server host, through the user host Connectivity Module, would query the user terminal for account data and a password. The user would enter the account data and password. This would be transmitted to the server host. At this point, the user terminal should be successfully logged into the server host and in a position to execute applications commensurate with his account at the server host.

The user should have the option of bypassing the selection menus offered by the Connectivity Module if he knows the correct symbols to enter for the server host he is interested in. Since almost all personnel in the supply and logistics community are intricately familiar with MILSTRIP RICs (Routing Identifier Codes), these could be used quite effectively since there is a correspondence between RICs and SPLICE nodes. The RIC is a 3 character alphanumeric symbol and most supply personnel have at least a dozen or so of the

most important ones memorized. For instance, if a user terminal at NSC Oakland, wants to connect with the server host at NSC Norfolk, he should be able to enter a command like:

CONNECT NNZ

or

CONNECT NSC NORFOLK

Entering a command such as CONNECT NNZ would obviate the need for experienced personnel to invoke the selection menu procedure. Entering a command such as CONNECT NSC NORFOLK would serve those who remember the common noun name of the activity they wish to connect with, but cannot remember the RIC offhand. Eventually, SPLICE will interact with stock points and ICPs outside of the Department of the Navy. The Connectivity Module should provide a table of RICs with DDN connections in addition to the abbreviated table of SPLICE nodes. Conceivably every RIC will eventually have capability for connectivity through DDN even if it is only a microcomputer hooked into a DDN TAC (Terminal Access Controller). A menu selection feature with UICs (Unit Identification Codes) is another possibility for identifying remote hosts, but would probably not have as much utility as a RIC menu until communication with hosts outside the supply community becomes prevalent.

Finally, the user terminal to server host connectivity module should have a menu driven selection for connection with server hosts outside the SPLICE community. Presumably, this would include hosts that are connected through the DDN but not part of SPLICE. The Connectivity Module should make the protocol of connectivity as painless and simple as possible for the unsophisticated user. All of the current hosts in the DDN could be loaded into a file that could be called by the Connectivity Module. The contents of this file would have to be loaded from data

provided by the NIC (Network Information Center) RFC (Request for Comments) titled "Assigned Numbers" which is periodically updated and promulgated for ARPANET host addresses [Ref. 23] and similar data maintained by DCA for MILNET host addresses. The user should then be able to browse the table containing all of the "Assigned Numbers" and pick the server host he desires connection with. This would again be accomplished via selection menu procedures implemented by the Connectivity Module. This table would necessarily be much longer than the SPLICE table and therefore should not be the default table for finding a remote host.

One method for implementing all of these table menus would be to set up a master menu that would query the user terminal to pick from:

- SPLICE Remote Host Table
- RIC (Routing Identifier Code) Remote Host Table
- UIC (Unit Identification Code) Remote Host Table
- Master DDN and ARPANET Remote Host Table

Of course, all of these tables should be loaded into one database or flat file that is indexed to produce the 4 selections outlined above. Other selection menu tables can be generated as required from this master address file. In fact, each individual user should be able to set up his own tailored table of hosts that he frequently communicates with, and have this tailored table be the default menu selection.

5. Requirements for Smart Asynchronous Terminal Connection to a SPLICE Host

The ability to effect transparent connectivity in this situation is more difficult than that of a terminal directly connected to a SPLICE node. In this situation, we assume that the terminal is connected into the DDN via a TAC

(Terminal Access Controller) or MINITAC, which is a smaller version of the TAC. Many of the selection menu processes which can be readily implemented on Burroughs bisynchronous terminals or IBM SNA SDLC terminals are not as easily handled in this environment. This is because most of the remotely connected terminals will be asynchronous. If they are dumb asynchronous terminals, the problem is even more extensive. Let us assume for the moment that we are working with smart asynchronous terminals, in the form of PCs (Personal Computers).

If we deal with a smart asynchronous terminal, then it should be possible to program a module within the Connectivity Module to support screen mode communications with the smart asynchronous terminal. Such a module would be downloaded from the SPLICE node to the smart asynchronous terminal as soon as connectivity is achieved. Full screen communication with a smart asynchronous terminal is complicated by the fact that normal communication appears in the scroll mode. This means that the current line appears at the bottom or 24th row on a standard screen, and pushes all data on the screen up one row. The top row of course disappears off the display when a new line appears at the bottom. Connection of these terminals directly into a DDN TAC or MINITAC is asynchronous, and this carries a high overhead per character transmitted, 25 to 37.5 percent. Such communication will not be as fast as that found on the bisynchronous and synchronous terminals directly hooked into the SPLICE Tandem machines but this does not rule out the utility of such connections. At 1200 bits/second or greater, it should be quite satisfactory, certainly a boon to any shipboard environment which in the past has never enjoyed access to such direct connections into stock points and ICPs. This application of course should be the bottom line for asynchronous connectivity with priority given to

the solution of the shipboard asynchronous connectivity problem. This is where the greatest benefit to material readiness and the Navy's fighting capabilities manifest themselves.

The most likely target for such an application would be the IBM PC and its clones that use the Intel 8088 microchip. The second priority target for such an application would be PCs that are based on the Zilog Z-80 and Intel 8080 microchip. Not all smart asynchronous PCs may realistically be supported, but certainly the IBM, Z-80 and 8080 PCs can and should be. The fact of the matter is that there is a large population of Zenith 100 and Zenith 120 PCs in the fleet, and these are Z-80 based machines. The current favorite for the fleet seems to be the newer Zenith 150, which is an IBM PC clone. The shore establishment is not that much different; again there is a mix of IBM PCs, IBM PC clones, Z-80 and 8080 machines. Since all Intel 8080 microchip programs will execute on the Zilog Z-80 microchip, we can basically reduce the problem to that of supporting two basic microchip architectures and compatibility with two PC operating systems. The IBM PCs and IBM PC clones use IBM's PC-DOS operating system, or Microsoft Corporation's MS-DOS operating system. The basic features of PC-DOS and MS-DOS are identical in function and will run interchangeably on either the IBM PC or the IBM PC clones. The Zilog Z-80 and Intel 8080 microprocessors run under Digital Research Corporation's CP/M operating system.

A basic monochrome terminal that doesn't support reverse images or highlighting should be assumed. Thus, some fancy features that are supported on more complicated screens will have to be sacrificed. This should not affect many applications. Right now there are very few if any bisynchronous or synchronous color terminals hooked directly into SPLICE Tandem machines, so color is really not an

issue. What reverse imaging and highlighting does exist may just have to be sacrificed in remote asynchronous communications. But this does not wipe out the utility of supporting full screen management with such hardware. Microchip programs that will conduct screen management are definitely possible. Witness the many PC software packages that employ menu facilities for prompting users through different applications. This same methodology can be employed for the remote smart asynchronous terminal.

The question is whether the entire screen management protocol and asynchronous communication protocol with the TAC can be handled in 64 kilobytes or less of memory. Programming of the screen management modules for the smart asynchronous terminals should assume a machine architecture with 64 kilobytes of RAM (Random Access Memory). If the limitation of 64 kilobytes of RAM (Random Access Memory) is too restrictive from a software engineering standpoint, perhaps the lower limit of smart terminal support should be 128 kilobytes of RAM.

An alternative implementation of full screen management, and one that may prove most expedient in the short-term, would involve the use of off the shelf PC software. These would include communication packages, simulator packages, and programs executed from floppy disk to emulate full screen modes found on the Burroughs bisynchronous and IBM SNA SDLC terminals. There are already a number of packages that provide asynchronous communication such as those marketed by IBM and Hayes. The problem is determining if they are compatible with direct hookup into Tandem hardware. We do know that these communications packages will allow connectivity into a DDN TAC. Other packages are marketed that will emulate the bisynchronous features of IBM 3270 terminals and IBM 3278 terminals. It may be just possible that one of these packages can be used

to emulate the Tandem terminals. If an off the shelf software package cannot be found that emulates the Tandem terminal, the next best thing would be to develop software for use on the PCs that would emulate the different menus and screen management facilities offered by the Tandem and Burroughs hardware for the bisynchronous terminals that are directly connected into the Tandem hardware. After a communications package such as Hayes Smartcom software is executing, and a connection is effected, the PC could be switched from terminal communications back to a stand alone PC. In this stand alone mode, a PC Menu Selection Module could be invoked that offers full screen management that steps unsophisticated users through processes such as "stock checks," "requisition status query," etc. The unsophisticated user would enter the data he is prompted for, making full use of "program function keys," the cursors, "page up" key, "page down" key, etc. All of these special function keys would be supported by the PC Menu Selection Module. Each menu should support a specific genre of transactions and should offer online data entry validation support. This online data validation support would typically do things like check whether a stock number entered has the correct number of digits, no alphabetic characters embedded in the wrong place, etc. The PC Menu Selection Module should have a bank of validation tables used to catch gross data entry errors for immediate online correction. Once the user has completed entering all the data for a particular class of transactions, he would indicate so to the PC Menu Selection Module, presumably by hitting one of the "program function keys." He may have only one stock check, or he may have entered 30. The PC Menu Selection Module will then transfer control to the PC Communications Module and send the transactional data entered in the stand alone mode as a file to the SPLICE

Tandem processor. The information received by the Tandem processor would be handled in a batch remote job entry fashion. The transactions would be processed and the results transmitted back to the user PC terminal. It is believed that handling transactions from asynchronous terminals in this manner is superior to basic online asynchronous transaction capability. Basic asynchronous transaction processing would not be user friendly, and would require the user to explicitly invoke all of the processes desired. This requires a high level of technical and systems expertise, a commodity rarely found in the fleet subject to constant turnover and influx of new personnel. Without the use of menus, templates, online help facilities, and user prompts, the benefits of SPLICE will be restricted to a small fleet audience. A PC Menu Selection Module would open the benefits of SPLICE to relatively untrained personnel, delivering critical real time information with minimum expenditure of skilled manpower resources.

6. Rationale for Nonsupport of Dumb Asynchronous Terminals by SPLICE

Although many "dumb" terminals abound throughout the Navy, further investment in the use of these terminals for SPLICE applications is not warranted. By a "dumb" terminal, we mean a terminal with no internal processing capabilities, and limited buffering capability. All keyboard actions must be transmitted to the host computer which interprets the data and controls the terminal. Since there has been such a proliferation of PCs throughout the Navy, it would be more expedient to concentrate on communications between SPLICE and PCs, which have their own storage and processing capabilities. Supporting the "dumb" terminal in an asynchronous environment is of very limited value due to its inherent limitations.

7. Requirements for SNAP I and II User Terminal Connectivity with SPLICE Hosts

The reasons for supplying SNAP I and II user terminal service into SPLICE are the same as with the smart asynchronous terminal. In fact the need is greater in this environment due to process-to-process interfaces between SNAP I and SNAP II processes and SPLICE. This process-to-process interface will be discussed later. The capability for supporting full screen management for SNAP I and II user terminals is greater than that found in the asynchronous stand alone PC environment. More efficient simulation of full screen mode is feasible which will support response times more in line with that expected from an online system. The major drawback to actual synchronous or bisynchronous terminal communications is the fact that connections into the DDN are asynchronous.

8. Requirements for Burroughs Bisynchronous and IBM SNA SDLC User Terminal Connectivity with SPLICE Hosts

As mentioned earlier in Chapter II, there is a need for transparent connectivity with IBM SNA SDLC user terminals which are being implemented at the ICPs, the Trident Refit Facilities, and CAIMS. Although these applications were not specified in the original Functional Description [Ref. 1] and System Specification [Ref. 3], it is clear that these applications are going to have an impact on SPLICE in the immediate future that is second only to the Burroughs bisynchronous communication problem. Until these user terminal compatibility problems are solved, user terminal to remote server host applications will be stymied. This is probably the most immediate and pressing problem to SPLICE implementation as of the time of writing.

B. ELECTRONIC MAIL FACILITY THROUGH SPLICE

One of the clear advantages of electronic mail is the speed by which textual traffic is transmitted, and the efficiency by which it is queued and processed by the receiver. In the logistics environment, the benefits would be great. An extraordinary amount of clerical time is wasted in the logistics environment trying to get a hold of a point of contact to get something accomplished or to transmit germane day to day operational information. Presently, this effort manifests itself in 3 basic ways.

1. Unreliability of Telephone Communication in the Navy

The most popular method of communication in the military is the telephone. But lack of reliable telephone service is probably the number one impediment to productivity in the Department of Defense. Even under conditions of perfect telephone service, the probability that a called party will be available for conversation is low. Time and space will not allow a digression into the frustrations of telephone communication while personnel are engaged in the day to day effort of running the Navy, in particular, the Navy logistics community. It is assumed that the reader finds these frustrations to be self-evident. This affects most negatively personnel at the grass roots level of clerical processes and middle management. GS-5 through GS-11 personnel frustrated in the accomplishment of their jobs due to inability to communicate with peers, slip into an unacceptably low level of productivity. This inability to communicate via telephone usually manifests itself in substitute forms of communication. For points of contact in close proximity, written or typed Memorandums are used. For personnel outside the local geographic area, Naval Letters and Naval Messages are often used.

The latter two forms of communication result in a high clerical overhead and cost. The letter is slow, and the message is often not received. While the message can generally be relied upon to transmit information within 24 hours, it is no longer an informal means of communication. Both the Naval Message and the Naval Letter carry the weight of official command correspondence. The chain of command in the process of chopping Naval Messages and Naval Letters cuts out pertinent information, and often inserts unwanted information. The original informal communication intended is not transmitted, doing further damage to the informal communication infrastructure that would allow our Navy to operate more efficiently.

2. Requirements for Electronic Mail

Electronic Mail packages are quite common. It is the most common way through which much technical information is passed between personnel in the research community through the ARPANET. It has and is working very effectively as a substitute for telephone conversation and written correspondence. It has the advantage of being very inexpensive, timely, reliable, and accurate. Multiple addresses as well as distribution lists can be utilized, resulting in the simultaneous and near instantaneous transmission of information. It has the distinct advantage over other written forms of communication in that the only clerical overhead is that incurred by the drafter. The SMTP (Simple Mail Transfer Protocol) [Ref. 24] adequately covers the technical details for an electronic mail specification. An excellent and powerful electronic mail software package that should be expanded upon to implement the SPLICE Electronic Mail Module is a facility called "MM" [Ref. 25,26,27]. MM has been implemented under the DEC (Digital Equipment Corporation) TOPS-20 operating system

[Ref. 28] and is widely used in the ARPANET community. MM offers the advantage of being in full compliance with SMTP specifications. Although MM offers excellent and powerful electronic mail processing facilities, it is currently geared for operation in the asynchronous mode of communication found in the ARPANET and DDN. Therefore, it operates in the scroll mode discussed earlier, requiring the user to explicitly type in and enter commands. an Electronic Mail Module should be developed using MM as a baseline. It should be enhanced to implement user friendly functions that step the novice user through to a successful transmission of electronic mail on his first attempt. This would include the use of template and formatted full screen capabilities to support multiple menus, online help facilities, use of program function keys, etc., as discussed earlier. The Electronic Mail Module should offer default use of a full screen user friendly word processing package. This word processing package should allow the novice to draft a clean and legible piece of correspondence on the first attempt without undue frustration. A package such as PEACHTEXT should be investigated for this function.

C. PROCESS TO PROCESS CONNECTIVITY

The physical implementation of SPLICE hardware at the Stock Points and ICPs will not simply result in interprocess communication within a closed Navy Stock Point and ICP environment. Since all SPLICE nodes will be connected through the DDN's MILNET, the potential exists for interprocess communication with any host in the MILNET system. This opens up doors that will allow interoperability of all logistic systems throughout the Department of Defense. In addition to this, it allows the interoperability of maintenance systems with logistic

systems. The possibility is real for true interoperability, coordination and communication between the Navy's maintenance processes and the logistic processes. To capitalize on these capabilities, farsighted decisions must be made that will ultimately support interoperability of systems that never before had a need for, or even contemplated communication with each other.

1. Requirements for SNAP I and SNAP II Connectivity with SPLICE

This category of connectivity exploits the distributed processing power between combatant vessels and SPLICE. Earlier the possibility of connecting shipboard PCs with SPLICE was explored. The reason this possibility came to fruition is that AT&T Dimension 2000 PBX (Public Branch Exchange) machines have been, or are replacing the old Stromberg - Carlson mechanical switch telephone exchanges on board ships. What this means is that finally shipboard voice grade lines have become reliable enough to support data communications. The Honeywell DPS-6 and Harris computers are the hardware for SNAP I and SNAP II respectively. With the Dimension 2000 PBX, it is now possible to link these machines directly into a Tandem mainframe when the ships are in port. This connection could facilitate online transactions between user terminals directly connected through a SNAP I or II user host, to a SPLICE server host. Again we have a situation where the Honeywell, Harris, and Tandem terminal management is probably incompatible. A software solution would have to be effected to allow for full screen management options offered to local Tandem terminals to be available also on SNAP I and II terminals. However, a distinct advantage to closing the loop between SNAP I, SNAP II, and SPLICE would be the ability to directly pass MILSTRIP transactions that

presently move through AUTODIN over telephone link or DDN TAC link to a SPLICE node. The advantages of doing so are clear. MILSTRIP requisitions and transactions could be automatically generated by the SNAP I and II hardware that is automatically transmitted to SPLICE nodes with no human intervention. The reverse flow of transactions would also be true, effecting a more real time update of logistic data bases on board ships.

These transactions would move more quickly through SPLICE and the DDN. Present arrangements for transferring MILSTRIP transactions from fleet combatants to the stock points include the following inefficient variations:

- Flying MILSTRIP transactions in the form of IBM 5081 and DD form 1348m punched card decks from aircraft carriers in the Mediterranean to the Naval Telecommunications Station at Sigonella, Sicily.
- Physical delivery of MILSTRIP transactions on magnetic tape to an activity with AUTODIN capability.
- Physical delivery of MILSTRIP transactions in the form of paper tape to an activity with a paper tape and AUTODIN interface.
- Transmission of MILSTRIP transactions via Naval Message to DAASO (Defense Automated Addressal Systems Office) only to be reformatted yet again for transmission through AUTODIN.

All of this is very inefficient, wastes fuel and manpower, and is a needless burden to the Naval Telecommunications message processing system. All of these methods are most assuredly slower and less reliable than packet switching via SPLICE and the DDN. When pierside, all MILSTRIP traffic should henceforth travel from ships to Stock Points via direct link to a SPLICE node, or indirect link to a SPLICE node via a DDN TAC. Use of Naval Message traffic for MILSTRIP transactions should never occur when ships are pierside. The reverse is equally true. All MILSTRIP traffic originating from Stock Points and ICPs destined for fleet units should be transmitted via SPLICE and the DDN whenever ships have a pierside connection.

Hopefully, at some point in the future, the MILNET will support satellite transmission of datagrams, allowing all logistic data communications traffic to flow through DDN and SPLICE while vessels are underway. Until this occurs, it is imperative that the Navy's maintenance and logistic communities present a united front to DCA demanding such capability of the MILNET.

2. Requirements for DLA AIS (Automated Information System) Connectivity with SPLICE

The DLA (Defense Logistics Agency) represents a significant source of both consumable and repair part material to fleet combatants. No discussion of global Navy material requirements can exclude or minimize the impact that DLA plays in supporting the fleet. Heretofore, system wide visibility of DLA material has been limited to major commands such as SPCC that enjoyed direct connections to SAMMS (Standard Automated Material Management System) databases at the 6 DLA ICPs. Since fleet material requirements supplied by DLA are so significant, it is imperative that interprocess operability with DLA be made a priority issue. Although there is some limited online visibility of DLA material, the entire contract administration mission of DLA is presently not visible to the Navy Supply System. A hefty percentage of all Navy contracts are being administered by DLA's CAS (Contract Administration Service). The actions of the DCASRs (Defense Contract Administration Services Regions), DCASMAs (Defense Contract Administration Services Management Areas), and DCASPROs (Defense Contract Administration Services Plant Representative Offices) are not visible at present, and there is no method or plan for interoperability. It behooves the Navy Supply System to gain access to MOCAS (Mechanization of Contract Administration Services)

processes if we are to improve our management of contracts in which we have vested interests. This is stated in light of alleged recent abuses in contracting, and the massive amount of news media exposure given to such occurrences. DLA is essentially an extension of the Navy Supply System and it is time that it is treated as its integral part, with an effort toward achieving fluid interoperability between DLA AISs and Navy Supply System AISs.

3. Requirements for Army, Marine Corps, and Air Force Logistic Systems Connectivity with SPLICE

The same reasons for requiring interconnectivity between SPLICE and DLA hold for interconnectivity between the sister services logistic AISs and SPLICE. Probably the biggest outside customer of the Air Force Logistic system is the Navy. This is due to management of aviation repair parts common to both services that the Air Force has cognizance over. SPLICE needs to communicate with Air Force logistic AISs if the Navy is to gain better visibility and control over aviation repair parts. Although not as significant, the requirement for material support from the Army and Marine Corps logistic systems are evident in support of day to day combatant vessel requisitions. There are a significant number of repair parts, particularly electronic ones that the Army has cognizance for. It behooves the Navy Supply System to gain visibility of Army material. The reverse arguments are also true. Each one of the sister services has a vested interest in gaining visibility of Navy material. Repair parts flow freely between the services in support of each others' respective missions. However, this flow is largely uncoordinated at the present time. Interoperability through SPLICE represents a vanguard effort to bring these separate logistic environments together for enhanced mutual material support.

4. Requirements for GSA Logistic AIS Connectivity with SPLICE

Often we take for granted and gloss over the vital role that the GSA (General Services Administration) plays in supplying the Navy with general consumable goods and tools. This support is not evident until such time that a vital consumable item cannot be procured due to some system-wide deficiency. When the paper shortage occurred several years ago, everyone was reminded of the role GSA plays in supporting fleet requirements, though the shortage was through no fault of GSA. Although they are not a member of the DoD establishment, a tremendous amount of material is supplied to the fleet by GSA. The GSA AISs are the final link that must be hooked into SPLICE to functionally view the Navy's material procurement process as a global system vice a maze of interrelated, but disjoint systems.

V. LONG-TERM REQUIREMENTS FOR TRANSPARENT INTERNETWORK CONNECTIVITY THROUGH SPLICE

The requirements cited in the previous chapter will simply enable differing user terminals and hosts on different systems to communicate with SPLICE nodes in a normal manner. In other words, the requirements basically deal with the problems of connection and translation, allowing one system to speak to another. This chapter addresses issues that take the concept of transparency quite a bit further. The true maturation of a distributed process lies in the ability to login to a terminal, and execute system-wide transactions without concern for the machine and the geographical location which is servicing the user requests.

A. AN ANALOGY BETWEEN EARLY 3RD GENERATION HARDWARE/SOFTWARE AND PRESENT STATE OF AFFAIRS

A rough analogy of where Navy Supply Corps AISs currently stand vis-a-vis the potential of mature distributed systems follows. Imagine that we are dealing with beginning third generation hardware and third generation languages. The tools provided us are far superior to the old solid state machines such as the AN/UYK-5(V). The use of a third generation language such as COBOL has released us from the necessity of knowing how data is handled internally (as was the case in assembler programming) and an operating system such as IBM's OS/MFT allowed us the luxury of multiprogramming. However, input functions, output functions, identification of files, location of files, detailed physical description of file

formats, their location, peripheral devices used, compiling, linkage editing, loading, job step executions, etc., must all be explicitly defined and stated. The user, programmer in this case, has to have intricate knowledge of the physical hardware setup as well as the software he is working with to successfully execute a batch job. Compare this scenario from not so long ago to that found on 4th generation hardware and operating systems, notably VM/CMS. Under such an operating system, you have what appears to be your own machine, you are no longer burdened with the physical details of peripheral devices and the storage of data. The programmer is liberated to concentrate on the problem of writing a correct program. The operating system assumes responsibility for the details of file storage and retrieval. All the user has to do is name the files, write them, and read them. Of course it is not quite so simple as portrayed here, but it is a quantum leap over the rigors of JCL and COBOL's Environment Division specifications.

We are still at the stage where detailed processes and machinery in the form of hosts must be explicitly stated. If it is a remote host, we must login to it before execution of any further processing.

B. FUNCTIONALLY RELEVANT DETAIL IN A DISTRIBUTED SYSTEM

Consider, if you will, how long American Airlines would be able to compete if their passenger reservation system required reservation personnel to identify explicitly which process at which location controlled the passenger reservation file for flight number 624 from Philadelphia to Detroit on 24 October 1985. Let us assume that there are several hosts in the AA reservation system, and that these hosts are distributed, as they most certainly are. The system would not last long because AA reservation personnel

would soon be relieved of the burden of such worrisome detail. United, Eastern and TWA would be doing it for them. Fortunately for AA, the reservation personnel need not concern themselves with such details. They are only concerned about functionally relevant data such as flight number, date, source of origin, destination, passenger names, etc. The system responds to these functionally relevant demands. It does not burden the reservation personnel by requiring them to identify explicitly the correct remote server host, execute an additional login procedure, specify the processes to be invoked to satisfy the functional request, etc. In large part, the beauty of such a distributed online system is a direct result of fierce competition between the airlines.

The BT3 (Boiler Technician 3rd Class) Supply Petty Officer for B Division on board an ADAMS class DDG doesn't have such a system to work with. Nobody is competing with the Navy Supply System, and it is the only show in town. Not quite, when the chips are down, his compatriots will abandon that system and opt for extensive and expensive repair of a valve vice replacement. Or his boss, the Chief Engineer, will get the local IMA (Intermediate Maintenance Activity) to buy off on a C-2 or C-3 CASREP and get a new part manufactured by one of the local commercial machine shops around the waterfront. The required parts may have been sitting in a bin down in the Supply Support Center, or sitting on a shelf at NSD Yokosuka, but they were never adequately identified. The parts may as well have never been procured and stocked by the Navy Supply System. As far as the BT3 and the Chief Engineer are concerned, they do not exist, and for all practical purposes, they don't! In either case, the Navy will pay dearly for parts that might have been identified, requisitioned, and delivered by the Navy Supply System in the same or shorter time than might be needed to overhaul or manufacture the parts.

Here are the relevant functional questions that any person directly supporting a maintenance and repair effort will have when a parts requirement has been identified:

- What is the part that I need?
- Where can I get it?
- How long will it take for delivery?

Due to time and space constraints, we will limit this discussion to the first functionally relevant question. Solutions to the second and third question follow the same line of reasoning presented below for the first question. In this generic shipboard situation, often the most taxing exercise in requisitioning a part is the basic problem of its definitive identification. For anything but the most routine and mundane identification problem, a substantial amount of technical expertise and minutia is involved. The problem is exacerbated if the ship's COSAL (Coordinated Shipboard Allowance List), technical manuals, and drawing numbers are out of date, incomplete, inaccurate, or worst of all, nameplate data from the failed part cannot be read or is unavailable. There are a variety of methods for definitive identification of a repair part, and a good SK1 (Storekeeper First Class) can proceed by several methods. Most all of these methods are time consuming, and there is no guarantee that a given ship has a sharp SK1 or SKC on board. It has been known not to be the case. Often, the the BT3 Supply Petty Officer for B Division will be more adept at identification of parts germane to B Division than storekeepers in the Supply Support Center. The BT3 may have been at this job for just under four years. He walks around with a logbook full of part numbers and National Stock Numbers that he painfully identified over the last three and a half years. B Division will greatly miss this BT3 if he gets transferred, leaves the Navy, or fails to pass on his logbook and his corporate knowledge before departure.

There is a systemic problem with parts identification and parts requisitioning. It is too labor intensive and too complicated for most supply personnel, let alone maintenance personnel, to learn proficiently. It is probably one of the most "user unfriendly" systems ever developed. Higher authority can insistently admonish that "A rigorous training program must be adhered to in order to overcome these technical difficulties and make the system work!" However, time and manpower resources are scarce commodities on board ship. The BT3 should have spent 4 years repairing equipment and learning his rate, not learning to become a storekeeper. How can we reverse this situation? The system must support user friendly global research capabilities for parts identification. This starts with hardware and software for SNAP I, SNAP II, and SPLICE sites in direct support of maintenance activities. For starters, the COSAL needs to be loaded into an online DBMS. Secondly, at a bare minimum, microfiche publications such as the ML-N (Management List Navy) and the MCRL (Master Cross Reference List) must also be loaded into a DBMS. a Parts Identification Module must be implemented which will step the novice user through to a successful identification of the required part. A storekeeper should not have to help him.

Let us assume for the moment that such a Parts Identification Module has been implemented in SNAP I and II. Such a system would require complex menu selection facilities and user prompts to extract as much information as possible that the user may possess about the problem at hand. Even the best expert system will not be able to identify all parts, particularly in cases when material is obsolete, no longer manufactured, or the ship's COSAL does not reflect a specific piece of equipment. The shipboard computer can only handle so much data. The three documents COSAL, ML-N, MCRL, are only a small subset of the technical

data available to identify parts. We must be able to go further through automation when SNAP I and II cannot handle an identification request. The request must be passed off through SPLICE to a suitable system or combination of systems that can satisfy the request. Powerful parts identification capabilities are available from the Supply Centers. Even more powerful facilities are available from DLSC (Defense Logistics Services Center) and SPCC (Ship's Parts Control Center). These activities by no means are the only tools available. The other armed services have their own capabilities which would be of great benefit when material under their cognizance is being researched. These capabilities do the BT3 no good unless he can invoke them.

Now let us further assume that there are sophisticated Parts Identification Modules available at the Naval Supply Centers, the ICPs and DLSC. DLSC used to support a rudimentary service whereby a part number requisition could be submitted via AUTODIN, and within 48 hours or so, DLSC would reply with an NSN (National Stock Number) that it successfully cross referenced. Obviously a much more sophisticated and general Parts Identification Module would be required, with response in the order of minutes if not seconds. We must do this through SPLICE and the DDN. The question is, how?

C. IMPLEMENTATION OF A DDS (DIRECTORY/DICTIONARY SYSTEM) FOR TRANSPARENT INTEROPERABILITY

To implement interoperability between modules located on differing systems at different nodes, the names and addresses of these modules must be identified. We assume that the request is handled in the same manner that the Navy handles its maintenance effort. We have a three tiered system starting on the bottom with the "Organizational"

level in direct support of the user, the "Intermediate" level which would handle requests beyond the scope of SNAP I or SNAP II, and a "Depot" level which would handle the most complicated requests by size and scope.

SNAP I, SNAP II, and SPLICE sites in direct support of maintenance effort should provide organizational level parts identification capability. Due to hardware and peripheral device constraints, the data base available for such processing will be limited in scope, perhaps to the three tools cited earlier. If the SNAP I or II Parts Identification Module fails in its search, it should then notify the user, attempt to gain additional information, and prepare to send the request to a remote server host possessing a Parts Identification Module with enhanced intermediate capabilities. Requests beyond the scope of a Parts Identification Module with intermediate level capability would be referred to a depot level Parts Identification Module.

1. A Process Resource DDS Implementation Scheme

In the most recent SPLICE report prepared by Schneidewind and Dolk [Ref. 29], the concept of a DDS is introduced. The most basic DDS should be implemented in SNAP I, SNAP II, and lesser SPLICE sites. In the particular case at hand, consider a situation wherein a SNAP I or II Parts Identification Module is unable to service an identification request due to lack of technical resources at hand. Given that the request is in the correct format, and has been validated as potentially solvable, the Parts Identification Module would "process resource fault" in the same manner as an operating system "page fault" in a virtual memory situation.

The term "fault" here does not indicate an error or a defect. The term stems from virtual memory operating

system architecture. In a virtual memory implementation, a "page fault" occurs when an executing process requires data that is not immediately available in RAM (Random Access Memory). The operating system must then fetch and read into RAM a "page" of data from secondary storage, typically a disk drive, before the process can continue executing.

A similar "fault" concept is presented here for distributed processing. If data required is not available locally, it must somehow complete the executing process by either fetching the information from some other node in the internetwork environment, or by transferring control of the entire process to another node possessing capability for complete execution of the interrupted process. The "process resource fault" would cause the Parts Identification Module to pass control to a Resource Fault Module. The Resource Fault Module would query the local DDS for information as to where to refer this "process resource fault" for further execution. The local DDS would provide this information and the Resource Fault Module would then execute an "internetwork call" to a remote server host identified by the local DDS as having capability for servicing the request at hand. Appropriate parameters would be passed to the server host process as part of the "internetwork call." In this case, a logical remote server host would be one of the Supply Centers. Presumably the SNAP I or II system would have some knowledge of the location where it is operating. In this manner, a request would not be directed to NSC Norfolk when the ship is located in Subic Bay.

But what happens when the local SNAP I or II DDS does not contain the information necessary for correct referral of a "process resource fault"? The situation is solved by the local DDS indicating so to the Resource Fault Module, while concurrently providing the address of a remote server host possessing an intermediate level DDS. State

information currently held by the Resource Fault Module is saved, and the entire request including the "process resource fault" state information is referred to a remote host with an intermediate DDS through an "internetwork call."

In the process resource DDS implementation scheme, the entire process of identifying a bogus part was "passed" via an "internetwork call" to a remote server host with a resident intermediate Parts Identification Module. In the case of a valid but more obscure request, the local DDS might not have the directory/dictionary information necessary to properly refer the request. In this case, the entire process of identifying the bogus part is "passed" via an "internetwork call" to a remote server host with an intermediate level DDS. This DDS will hopefully contain the directory/dictionary data necessary for proper referral of the request. When the intermediate Parts Identification Module "process resource faults" for lack of processing capability, the request is referred to a depot level Parts Identification Module. If the remote server host DDS or an intermediate level DDS cannot properly refer the "process resource fault," the entire process is referred to a depot level DDS. The relationship between organizational processes/DDS and intermediate processes/DDS are analogous to intermediate processes/DDS and depot processes/DDS.

2. Physical Location of Hierarchical DDS

Schneidewind and Dolk offer two extreme alternatives as to degree of distribution for the DDS [Ref. 29: p. 65, 66], as well as a compromise implementation. Herein, it is proposed that a three tiered hierarchical DDS also be implemented. The most rudimentary DDS shall be installed in all SNAP I, SNAP II, and minor SPLICE site installations. Intermediate level DDSs should be implemented at NSC Pearl

Harbor, HI; NSC Oakland, CA; NSC Norfolk, VA; and NSA Naples, Italy. The intermediate level DDSs should be able to handle all directory/dictionary referral requests that cannot be handled by the organizational level DDS at SNAP I, SNAP II and lesser SPLICE sites. Online storage requirements and activity level will determine if intermediate level DDS facilities should be implemented at all of the Naval Supply Centers. The intermediate level DDS should be able to direct all requests that potentially can be serviced by a module somewhere within the confines of the Department of the Navy. Depot level DDS facilities should be located at the ICPs, SPCC and ASO (Aviation Supply Office) respectively. These depot level DDSs should be capable of directing all requests outside the purview of the Department of the Navy, to modules resident at remote server hosts offered by DLA, sister services, and GSA. The DDS hierarchical demarcation lines are very arbitrary at this point. They can and will be restructured to fit the situation as it develops during implementation.

The rationale for location of intermediate DDS facilities for such modules is to service each Fleet. Although NSA Naples is not is considered a major stock point, it represents the hub of all Sixth Fleet logistics. To refer all Sixth Fleet directory/dictionary referral requests to, say, NSC Norfolk in CONUS is considered to be unacceptably inefficient. Perhaps the DDS located at NSC Pearl Harbor should be moved closer to Seventh Fleet action, say, NSD Guam. Furthermore, it may prove expedient to install an intermediate DDS facility at Diego Garcia, BIOT. In any event, what is envisioned is a hierarchical DDS architecture. When a lower level DDS is incapable of directing a "resource fault" to a remote server host, the lower level DDS will pass parameters indicating this to the local Resource Fault Module, and specify the address of the

next higher DDS facility in reasonable proximity. It should be emphasized that the intermediate and depot level DDS host in most cases will not have the capability for handling a user host "resource fault." Higher level DDSs provide detailed maps of system resources. The resident Resource Fault Module at the intermediate or depot DDS uses this information as well as the "resource fault" state information passed to it from the user host, and directs the "resource fault" to an appropriate remote server host through an "internetwork call." In this particular case, we are interested in only the location of hardware, software and data required for satisfying the "resource fault." The total content of what the DDS offers is much larger in scope, and the reader is referred to Schneidewind and Dolk [Ref. 29: pp. 58 thru 64].

3. A Data Resource DDS Implementation Scheme

One of the fundamental assumptions in the organizational level Parts Identification Module scenario is that the entire process that "process resource faults" is kicked up to the next higher level server host process via an "internetwork call." The user host simply waits while higher level processes at remote server hosts take control of the entire request and execute it to completion. When process execution is completed, the results are transmitted back to the user host. It is not efficient for SNAP I or II to maintain control over a process that has "resource faulted" for lack of data. The cost of communication overhead between a shipboard SNAP I or II process and data resources scattered throughout the system would be too great. Therefore, the SNAP I or II system will "process resource fault" whenever required data is not locally available, regardless of whether local processing capability exists for execution to successful completion. The Tandem

TXPs at SPLICE sites have much more processing power, have superior data communication lines and facilities, and they are designed for transactional processing. It is assumed that the SNAP I and II sites do not possess the processing capabilities or the high speed data communication connections that will be required for more complex processes. The data communication lines between SNAP I and II sites must be assumed tenuous at best, and can be broken at any moment due to operational necessity or by the very nature of the pierside environment. If an entire request is referred to a higher level process ashore, then even if connection is broken unexpectedly, results can be spooled and transmitted to the ship at such time that data communications are reestablished.

Let us now dissect the DDS scheme from a data resource point of view. In this situation we will assume that an intermediate level Parts Identification Module at NSC Jacksonville is handling a process referred from the USS FORRESTAL (CV 59) which "process resource faulted." In this example, the intermediate level Parts Identification Module will be in control throughout execution of the request to a successful conclusion. It will not "process resource fault" to a depot level Parts Identification Module. However, in the process of servicing this request, the intermediate Parts Identification Module will "data resource fault" for data not locally available. For the sake of argument, let us assume that it requires a copy of APL 841800019 and the piece identification table included in NAVSEA drawing number 805-1349742 revision 3. Thus two "data resource faults" will occur.

The first "data resource fault" will occur to extract a copy of APL 841800019 from the WSF (Weapons System File) at SPCC Mechanicsburg. The intermediate Parts Identification Module would signal a "data resource fault"

to the resident Resource Fault Module. The locally resident DDS would indicate to the Resource Fault Module, that the information required is in the WSF at SPCC. The Resource Fault Module would execute an "internetwork call" to the SPCC SPLICE complex requesting a download of that particular APL from SPCC. The file containing that APL is successfully downloaded, and the "data resource fault" that precipitated this downloading process is satisfied. In the future, each Supply Center may actually hold an online copy of all APLs, but for the sake of illustrating this point, we assume that a copy must be downloaded from SPCC.

In the second "data resource fault" situation, the "data resource fault" is passed to the resident Resource Fault Module. The Resource Fault Module queries the local DDS to see where NAVSEA drawing number 805-1349742 revision 3 can be found. The local DDS indicates to the Resource Fault Module that it cannot generate that data. It indicates to the Resource Fault Module that it should direct this query to the intermediate level DDS located at NSC Norfolk. The Resource Fault Module executes a "data resource fault" "internetwork call" which includes pertinent state parameters to the DDS resident at the NSC Norfolk SPLICE complex. The NSC Norfolk Resource Fault Module takes this request and executes the query against the intermediate level DDS. The intermediate level DDS is also unable to generate a solution. Now the highest level DDS "data resource fault" occurs. The NSC Norfolk intermediate DDS indicates to its resident Resource Fault Module that it must pass the query to the depot level DDS at SPCC Mechanicsburg. So the NSC Norfolk Resource Fault Module executes an "internetwork call" to the SPCC Mechanicsburg depot level DDS, concurrently passing relevant state information about the user request process that originally "data resource faulted" at NSC Jacksonville. The SPCC Mechanicsburg

Resource Fault Module takes this request, and queries the depot level DDS for a solution. The depot level DDS indicates that the information required is located online at Long Beach Naval Shipyard. The SPCC Resource Fault Module then executes an "internetwork call" to the SPLICE site at Long Beach Naval Shipyard directing it to transfer the file that contains the piece identification table for NAVSEA drawing number 805-1349742 revision 3, to the process in the user host at the NSC Jacksonville SPLICE site that "data resource faulted" for this information.

The intermediate Parts Identification Module at NSC Jacksonville now has both "data resource faults" satisfied, and is now in a position to resume execution. Happily, it finds that NSN 7G 4320-00-237-4861 is the solution to the parts identification query originally passed from the USS FORRESTAL (CV 59). The NSC Jacksonville Resource Fault Module then executes an "internetwork call" to pass this information back to the SNAP I process on board FORRESTAL which generated the "process resource fault" that started this procedure.

4. Implementation Considerations

The process and data "resource fault" procedures cited above illustrate a possible implementation for transparent execution of complex processes requiring system wide resources in dispersed and varying locations. The author does not intend to imply that the processes cited above are the correct technical method of solving the problems illustrated. Such is a systems analysis problem. What is intended, is to focus on a general methodology for execution of such transactions. Obviously, in some situations, a DDS will respond that requested data is simply not available, or that data is available, but in hard copy form. Such a scenario would require human intervention and

manual review. Early implementation of a Parts Identification Module would be fraught with "data resource faults" where the DDS indicates that the data is not online, but is in hard copy format. However, until such transparent systems modules are implemented, it will be difficult to identify all logistic data that should be digitized and online. Not everything can be brought online at once. It is an evolutionary process. Once such systems are up and running, the most critical data not online will quickly be identified for digitization. Depending upon the priority of the requisition, a threshold for manual review would have to be established. It is recognized that there is potential for a lot of abuse of such a system through users unwittingly causing massive and complex searches to be executed. The user interface and validation modules for a Parts Identification Module must attempt to screen out such requests and limit processing to valid and potentially solvable problems. Through similar implementations, the other two functionally relevant questions could also be transparently automated. Of course, there is no limit to the number of processes that could be supported. The ultimate criteria for deciding which applications to implement first will be projected return on investment measured by that universal yardstick, improved material readiness on board ships. It is the author's belief that work should commence to design such internetwork transparent processes to exploit the benefits of SPLICE and to bring a user friendly revolution to the end users of the Navy Supply System, the BT3, the AMS2, and their compatriots.

VI. REFERRAL OF MILSTRIP REQUISITIONS IN A DISTRIBUTED ENVIRONMENT

One of the problems presented by radical changes in automation is a realization that the most efficient method of accomplishing an objective is not simply to streamline and automate a manual system. Sometimes, it is necessary to take inventory of the tools and resources available, reassessing the best method for getting from point A to point B with these new tools. Since we are in the dawn of a distributed processing revolution in the Navy Supply Corps, it is time to take a look at the fundamental process of MILSTRIP requisition processing. In particular, the MILSTRIP referral process. At the present time, when an end user submits a requisition to the local stock point, several things can occur. These actions are summarized below.

- The UADPS-SP process checks to see if the NSN in question is carried locally in stock.
- If the NSN is carried, and the quantity on hand meets the demand, a material release order is cut to satisfy the requisition.
- If the material is not carried, or there is not enough stock on hand to satisfy demand, the requisition is referred to the ICP.
- The ICP examines the requisition, checks to see where this particular material is stocked, and once again refers the requisition to an appropriate stock point believed to have on hand stock of this NSN.

In the event of a referral to the ICP, this referral process from the stock point to the ICP and vice versa often takes on the characteristics of a recursive function as the ICP vainly searches for a stock point that carries the material demanded. Why does this recursion in the referral process occur? Because the ICP does not possess real time information about the inventory level at the stock points, or the inventory records that the stock point reflects are inaccurate.

Normally these requisition referrals traverse through ICPs and stock points via AUTODIN. So the process is not exactly what could be referred to as occurring with blinding speed. Anyone who has spent any time tracking MILSTRIP status knows that on the average, it takes at least a day for a requisition, even if it is Issue Group I, to traverse an ICP or a stock point during the referral process. It doesn't have to be this slow. With SPLICE and the DDN, there is no longer a fundamental reason for referring all requisitions to the ICP when frustrated at a stock point. Through packet switching, it is possible to refer requisitions from one stock point to another with great speed. The old line of reasoning is that the ICP is the only organization that has the overall visibility of stock throughout the Supply System. Therefore, it appears logical to refer all frustrated requisitions to an ICP for proper routing. This made good sense in a 1967 batch processing environment. Thus we have centralized management of resources. This argument no longer holds true today. The Tandem NonStop TXPs are extremely powerful transaction machines. The capacity of disk storage has increased tremendously since the IBM 2314 disk drives, and the cost per byte stored has fallen drastically. There is no reason that the NIR (National Item Record) cannot be loaded into disk storage at every major stock point, certainly all the NSCs and NSDs. It could be set up in a very simple indexed format illustrated by Figure 6.1. There is of course no physical limitation to implementation of an index along the same lines as that illustrated in Figure 6.1. Every major stock point should be capable of intelligently referring requisitions without ICP intervention if such an index is loaded online.

The NIR Referral Index is indexed by NIIN (National Item Identification Number). For each NIIN, The RICs (Routing

NIIN	RICs	ICP
002413891	NNZ NOZ NDZ	GSA
002413893	N23	N35
002413894	NNZ NPZ B16	N32
002413895	NNZ NOZ NDZ NPZ S9G	S9G
002413897	NOZ NNZ	N35
002413898	NNZ NOZ NDZ NPZ	S9M

Figure 6.1 Example of a Stock Point NIR Referral Index.

Identifier Codes) would be listed in descending order of probability of successful referral. This means that the stock point with the greatest stock on hand would be first on the list, and those stock points with only one each on hand would be listed at the end of the RIC list. A referral algorithm based upon probability of successful referral, proximity, and other factors could be developed quite easily to govern rules of referral.

A. IMPLEMENTATION OF A STOCK POINT NIR REFERRAL INDEX

Let us assume that all of the NSCs and NSDs implement such an index. When a requisition is frustrated at the stock point of origin, SPLICE will package the requisition in a TCP (Transmission Control Protocol) segment for referral to the next logical stock point indicated by the NIR index and routing algorithm. At the end of the data segment, behind the MILSTRIP data, the RICs of all stock points traversed by this MILSTRIP document would be appended. Thus each MILSTRIP transaction now carries its history of traversal with it. By examining these RICs before further referral, the next stock point will not cause the requisition to be referred back to a stock point that it has already visited. Thus, the recursive referral problem

is avoided. After an arbitrary number of RICs are appended to the end of this TCP segment, perhaps 8, SPLICE would then automatically refer this requisition to the appropriate ICP for further processing, or manual review. The ICP column in Figure 6.1 can of course be substituted with the COG (Cognizance Symbol). If the COG is used vice the ICP RIC, another index would be required to translate the COG to a RIC prior to transmission. This is an implementation detail.

B. ADVANTAGES OF IMPLEMENTING A STOCK POINT NIR REFERRAL INDEX

There are three major advantages to implementation of such a referral scheme. First of all, this method of dynamically routing requisitions between stock points relieves the ICP computer systems of a tremendous workload. At the same time, the workload at the SPLICE sites is increased only marginally.

Secondly, transmission of MILSTRIP documents by packet switching through the DDN is much faster, more reliable, and more efficient than transmission via AUTODIN. The end result is movement of material to the fleet in a more timely fashion.

Thirdly, such an implementation would foster much needed redundancy throughout the Navy Supply System. Presently, operation of the entire Supply System is hostage to the referral processes at SPCC and ASO. In a wartime environment, this could prove disastrous. Using the distributed processing tools offered by SPLICE, and the redundancy built into the DDN, we can break the yoke of such a centralized and vulnerable mode of operation. If the NIR index were implemented at all of the major stock points, ASO and SPCC could both be destroyed, but MILSTRIP referrals

would continue to traverse through the Supply System via SPLICE and the DDN.

C. DISADVANTAGES OF IMPLEMENTING A STOCK POINT NIR REFERRAL INDEX

The obvious disadvantage to implementation of a NIR Referral Index is the cost of disk storage space to implement such an index. In order to make it truly redundant in a wartime environment, an abbreviated copy of DLSC's (Defense Logistics Services Center) NIR should be loaded on disk. If it were intelligently formatted, it would likely require anywhere between one and two 512 megabyte disk drives. Obviously there is more processing overhead involved at the stock points in looking up each frustrated requisition in the NIR index. However, the Tandem NonStop TXP machines were designed for such functions, so it should not be much of a problem. Periodically, the ICPs would have to promulgate changes to the NIR index, so that all of the stock points are using uniform and current information. The major overhead would be in initially loading such an index, maintenance of it should prove not very taxing. As a final note, special care must be taken to implement the "TCP Quiet Time Concept" discussed in Appendix B. This special TCP implementation must be adhered to in order to avoid duplicate MILSTRIP transactions traversing the Supply System.

VII. SPLICE INTERPROCESS CONNECTIVITY IMPLEMENTATION ISSUES AND CONCLUSIONS

In the previous three chapters, we have seen how SPLICE will fundamentally change the way we shall conduct business in the near and long term. Here, some implementation issues are raised.

A. EXPAND/X.25 VERSUS TCP/IP DDN IMPLEMENTATION

Chapter II mentioned DDN's current inability to support IBM SNA SDLC terminal communications and Burroughs Bisynchronous Poll/Select terminal communications. A possible solution to this problem is the implementation of Tandem Corporation's off the shelf EXPAND communications software which uses the services of the CCITT X.25 standard. For a detailed explanation of the CCITT X.25 standard, the reader is referred to Appendix A. In DDN parlance, this CCITT X.25 standard is referred to as "Basic X.25." DDN will soon implement an X.25 version that will adhere to military specifications. This militarized X.25 implementation is referred to as "Standard X.25." Tandem's EXPAND software communications package runs on top of "Basic X.25." "Basic X.25" service will be supported by the DDN until at least 1 October 1988, but soon thereafter it will support only "Standard X.25" service [Ref. 30]. Through the use of other utility programs offered by Tandem, namely SNAX and SNAFU, supposedly these synchronous and bisynchronous terminal incompatibilities with the DDN can be solved.

The logical solution for terminal communications would be the implementation of TELNET (Network Virtual Terminal Protocol) [Ref. 31] which rides above TCP/IP (Transmission

Control Protocol/Internet Protocol) [Ref. 32,33,34]. Appendix B contains a detailed discussion and overview of TCP/IP implementation. The stumbling block to implementation of TELNET is the fact that it only supports asynchronous terminal communications. As stated in Chapter II, DCA intends to implement an IBM 3270 bisynchronous protocol, but no plans are made for implementation of IBM's SNA SDLC. This is a most unfortunate situation. Regardless of DCA's stand on this issue, IBM SNA SDLC is rapidly becoming the industry de facto standard. Ignoring this fact only handcuffs current IBM SNA SDLC implementations in the DDN, and sours any future desire by DoD customers to implement the DDN. The DDN is the DoD mandated common switched data network. It is a monopoly service for DoD. Therefore, the users have no choice but to put up with whatever inconveniences it causes. It is the author's concern that delays in delivering compatibility with current commercial industry standards may eventually lead to substandard service for DDN users. A change in the DCA policy with respect to the IBM SNA SDLC issue will send a signal to its customers that it intends to continue supporting innovative commercial services about to become industry standards. DCA's current position is that in addition to TELNET, it will support IBM 3270 BSC (Binary Synchronous Communications) DSP (Display System Protocol) [Ref. 30], but will not develop support of any other synchronous protocol. As was mentioned in Chapter II, bisynchronous communications are being rendered obsolete by synchronous communications. DCA's present support for IBM BSC DSP is sound, but the author fears that this is becoming an aged technology. The DCA, as this author sees it, should stay ahead of its customers' demands rather than react to them. It is recognized that implementation of the DDN is a tremendous task and that there are conflicting priorities in

the overall scheme of implementation that severely tax the financial and personnel resources of DCA. However, the burden should not be placed upon the customer for development of an IBM SNA SDLC protocol specification. This is what DCA is currently requesting from NAVSUP in response to demands for IBM SNA SDLC support. The general view is that IBM SNA SDLC will become widely accepted. The author feels that DCA should react accordingly. We are not dealing with some unique protocol that some lesser organization developed. Recognition must be given to the fact that in this instance IBM is again setting an industry standard. Delays in giving support to this quickly emerging standard for terminal communications may be a disservice to DDN customers.

As a result of this IBM SNA SDLC situation, NAVSUP is seriously considering implementation of EXPAND over "Basic X.25" concurrent with an implementation of TCP/IP. This an acceptable stopgap measure, but only until such time IBM SNA SDLC is supported by the DDN. Use of TELNET or a DDN standard protocol is much more desireable than locking SPLICE into use of EXPAND/X.25. Before such a move is executed, NAVSUP should take a very serious look at just what its IBM SNA SDLC user terminal to remote server host demands will be. If it cannot identify some very urgent application requirements that would be actively used the day such service became available, perhaps it should instead push for a DDN IBM SNA SDLC protocol standard. If application programs have yet to be developed before user terminal to remote server host communications become a necessity, then NAVSUP should definitely wait the situation out and lobby hard for a DDN IBM SNA SDLC solution. Although concurrent implementation of EXPAND/X.25 and TCP/IP sounds inviting, it is fraught with software engineering difficulties.

B. SOFTWARE ENGINEERING QUALITY ASSURANCE FOR DDN INTEROPERABILITY

The most significant problem with implementation of EXPAND lies in the danger that application programs may be developed that will be coupled to EXPAND internal software, an undesireable software engineering situation. If this happens, applications software will inextricably be tied to Tandem Software and consequently Tandem Hardware. SPLICE will foster a variety of new applications software. Every effort should be made to ensure these application modules interface with DDN standard higher level protocols, specifically TELNET (Network Virtual Terminal Protocol) [Ref. 31], FTP (File Transfer Protocol) [Ref. 35], and SMTP (Simple Mail Transfer Protocol) [Ref. 24]. By pursuing such a course of action it will be possible to reap the benefits of interoperability with other diverse logistic and nonlogistic systems in the future. An acceptable interim solution to this problem would be development of server modules that provide TELNET, FTP and SMTP specification interfaces for applications. How Tandem chooses to make these server modules work would no longer be that important. The services of TCP/IP or EXPAND/X.25 could be used by the server modules and standard interface integrity of application modules would be maintained. With standard TELNET, FTP and SMTP interfaces, all application modules developed would be assured of interoperability with replacement hardware that offers the DDN protocols. Through use of the higher level DDN protocols and their interfaces, SPLICE applications would be insulated from coupling with Tandem or any other vendor-specific software. The Navy would not be dependent on a sole source vendor when time comes for resolicitation and replacement. Ensuring smooth software interfaces with DDN protocols must be the

overriding criteria in choosing an end-to-end reliable connectivity plan. For reasons cited in the three previous chapters, it is crucial that all software from this point forward be developed with standard DDN protocol interfaces. SPLICE has an obligation to ensure that interoperability with SNAP I and SNAP II can be effected smoothly. These requirements are starting to appear on the horizon. NAVMASSO (Navy Management Systems Support Office) will certainly not implement EXPAND/X.25 for a SNAP I or SNAP II interface with SPLICE. The interface will often be through direct communication link from SPLICE server host to the ship user host, but this will not be the case when a ship pulls into any of a number of foreign ports. DDN will service such data communications, so DDN protocols will be required. Such is the only method by which the Navy, let alone DoD logistic communities, can ever hope to avoid the "Tower of Babel" syndrome.

If standard DDN protocol software interfaces are not developed and utilized, there will be little hope for achieving true interoperability when the rest of the Navy and the sister services begin implementing their distributed systems during the next decade. The Navy Supply System has a vested interest in interoperability. DLA or GSA will not suffer if there is no interoperability since they are not the Navy's customers. The Navy is their customer. Likewise, the Navy is also a customer of the sister services and vice versa. The foundation must be built now for interoperability with logistic as well as nonlogistic AISs that will follow SPLICE's debut on the DDN. This will be difficult to achieve if SPLICE application programs are developed or modified with EXPAND interfaces.

C. RAMIFICATIONS OF CONCURRENT IMPLEMENTATION OF EXPAND/X.25 AND TCP/IP

Concurrent implementation of two end-to-end reliable interconnectivity protocols at the same time can only result in software engineering as well as performance problems. To begin with, two distinct sets of protocols will have to be maintained which will cause compatibility and configuration problems for whoever is responsible for maintenance. It will be akin to tampering with and attempting to implement two concurrent, completely different operating systems. Witness the performance degradation that takes place when IBM's VM orchestrates the concurrent operation of CMS and MVS. Eliminate VM and one of the two operating systems, and system performance for the single remaining operating system is dramatically improved. EXPAND cannot serve as the single network interface "operating system." It will result in a closed community which is unacceptable from an internetworking standpoint. In such a situation, few processes discussed in the previous three chapters could be implemented. Concurrent implementation of two "internetwork" operating systems will be a software engineering nightmare. It will result in a lot of wasted software effort contrary to the principles of sound software engineering [Ref. 36,37,38].

FMSO cannot afford to get involved in supporting two plans for internetwork operability. It has a hard enough time with its current workload. It is better to remain with the standard DDN protocols and forego the extra features of EXPAND. The whole concept of SPLICE revolves around standardization of interfaces and relief from different patchwork communications schemes currently handled by a myriad of hardware and software. Concurrent implementation of TCP/IP along with EXPAND/X.25 may bring back serious

configuration management problems from which the Navy Supply System has been trying to liberate itself.

APPENDIX A

COMMERCIAL PACKET SWITCHING INTERNETWORKING METHODOLOGY

In view of the DDN mandate cited in Chapter II, it is assumed that the DDN will be the backbone of the SPLICE wide area network. For this reason, packet switching methodologies will be emphasized herein. Other major long haul data communications methodologies include use of dial up telephone lines, dedicated lines, commercial packet switched networks such as Tymnet or GTE Telenet, or a combination of these options. A question that remains to be answered is which sets of protocols and methodologies will be used to implement this wide area data communications.

A. DATA COMMUNICATIONS SWITCHING METHODOLOGY

Data communications is basically broken down into three different methodologies of transmission.

1. Circuit Switching

Circuit switching is in all practicality the same type of connection that is made by two individuals when talking on the telephone. Physical resources in time, space, or frequency spectrum are dedicated to the exclusive use of a single call for the duration of that call [Ref. 4: p. 27]. Different switches connected to each other throughout the network create what is logically and for the most part physically equivalent to a twisted wire pair connection. The advantage to such a connection is that, as long as one is willing to pay for and support the connection, it is at the user's disposal with no interference from other users contending for the line. Of

course the disadvantage to such a system in computer-to-computer communications is the cost of doing this over long distance. Imagine the cost of a continuous computer-to-computer telephone connection between Boston and San Francisco, 24 hours a day. In the absence of constant traffic over such a connection, it would quickly become economically prohibitive for all but the most critical situations. Of course the users or operating systems of two computers always have the option of dialing up the connection when the operating system of one of the computers has a need to communicate with its distant counterpart. But then line contention could be a problem, quality of the connection may be suspect, and a dialup delay may be deemed unacceptable for the application in question. This is an oversimplification of circuit switching methodology, but for the purposes of this discussion it is a traditional telephone connection with a physical end-to-end hard wire linkage. Information is transferred with negligible delay. A short but more formal definition is offered, "A form of switched network that provides an end-to-end path between user endpoints under the control of network switches" [Ref. 4: p. 356].

2. Message Switching

In message switching, each switch in the transmission path of a message successively stores the transmitted message in its entirety. Messages are stored on a permanent medium such as magnetic tape or magnetic disk for a specified duration of time, typically on the order of one month. If the connection to the next switch is open and the line is not busy, then the message will be forwarded down the line. If the next connection is busy or down, the message is delayed. When an appropriate connection path is free, the message will be forwarded to the next switch in

the system enroute to final destination. All intermediate switches that a message passes through hold a permanent copy of that message on file. Thus, message switching is commonly referred to as a "store and forward" system. The system works very much like a TWX, Western Union telegram or Naval message. In fact, these systems incorporate message switching technology. This type of transmission is very economical. Message switching can use much narrower transmission bandwidths than voice communications, so accurate transmission is possible on poor quality communications circuits [Ref. 4: pp. 28, 29].

Since each message must be transmitted in entirety, long messages in the system can and often do block transmission of other shorter messages. Thus it is not an ideal system for real time, interactive or time sensitive communication. From a military security point of view, it has the unattractive feature that an entire message is stored at several intermediate switches between the sender and the receiver. Thus the entire message contents are subject to compromise.

3. Packet Switching

Packet switching is a special case of message switching. The maximum message length is severely restricted, so you don't have the blocking problem that large messages may cause in a traditional message switched network. The packets move through the network, working their way toward their final destination on a hold and forward basis. Each switching center holds the packet temporarily until it is sure that the next switch down the line properly received the packet. Unlike message switching, no permanent record of the transaction is held. When the sending switch has confirmation of packet receipt from the receiving switch, the sending switch is free to

dispose of the packet. Packet switching has most of the advantages of a message switched network. In addition, long messages and short messages do not interfere with each other, thus providing rapid and efficient throughput except under extreme overload conditions. Therefore, it can be used in an real time, interactive or time sensitive environment. One of the greatest advantages to packet switching, in addition to its speed, is its flexibility. The packets do not have to follow a prespecified path to their destination. Their routing can be adapted between switches in the system depending upon traffic load patterns. Packets can be routed around switch failures in the system. In most cases, neither the sender, nor the receiver need be concerned with individual switch failures. To achieve this flexibility, adaptability and speed, many small switches and processors (computers) must be implemented. This in turn requires complex routing and control procedures. The packet switched network has rapidly emerged as the long haul data communications network of choice. It is the subject and foundation for this research effort [Ref. 4: pp. 30, 37, 38].

B. PACKET SWITCHING ALTERNATIVES

In Chapter II, packet switching was introduced. What follows is a discussion of two different implementation schemes for packet switching.

1. Datagram Approach

The network treats each packet independently in the datagram approach. Suppose that a file is to be sent from a host in Boston to a host in San Francisco. Let us further suppose that this file is of such a size that it will be divided into 5 packets. The packets will be transmitted

from Boston in sequence: 1, 2, 3, 4, 5. However, the routing of an individual packet once it reaches the first node or "switch" is independent of the routing for other packets comprising the file being transferred. Each switch in the system makes a decision as to how best to route an individual packet toward its final destination. Typically, these routing decisions are made on the basis of some optimum routing algorithm which changes dynamically according to the overall status of the network. So for packet 1 to traverse the distance from a host in Boston to a host in San Francisco, it may travel by way of a node in New York City, then Atlanta, then Dallas, then Denver, and finally San Francisco. Packet 2 may travel by way of Chicago, then St. Louis, then Salt Lake City, and finally San Francisco. When the network is heavily loaded between two switches, or if a switch is detected to be dead, there are reasons for routing a packet on what may seem a circuitous route. If the Chicago switch is overloaded, there is reason for routing packet 1 to New York City and Atlanta. The algorithm that a switch employs for optimum routing of packets does not concern us. What is important to remember is that as a result of this method of transmission, packets are likely to arrive at the San Francisco host out of sequence. Their arrival sequence might be: 1, 3, 5, 2, 4. Thus it becomes necessary to reassemble the arriving packets properly in order to recreate the file transmitted from Boston.

The information contained in each datagram is useful by itself to the end user. It does not depend on the contents of preceding or following datagrams in order to be of utility to the destination host [Ref. 4: pp. 117, 121]. Each datagram is fully identified as to destination and sequence number within a transmission. One major advantage of the datagram approach is that there is no need for call

set up. If a host needs to send only one or a few packets, datagram delivery will be quicker than the virtual circuit approach [Ref. 8: pp. 31, 32]. Secondly, since datagram service is more primitive, it is more flexible. Finally, datagram delivery is more reliable. If a switch in the normal path of routing fails, the neighboring switches simply route packets around the failed switch. Of course, an obvious disadvantage to datagram service is the requirement for the receiving host to successfully reassemble packets that are out of sequence and obtain retransmission of missing packets.

2. Virtual Circuit Approach

In virtual circuit service, a logical connection between Boston and San Francisco would be set up before transmission of any packets. The Boston host would initially send a "call request" to the San Francisco host. The call request might be routed first to Cleveland, then to St. Paul, then to Spokane, and finally to San Francisco. A virtual circuit has now been established. For the duration of this connection all packets will travel by way of Cleveland, St. Paul, and Spokane. Any packets or acknowledgements transmitted from San Francisco to Boston will travel the reverse route. The important point to remember is that once the call request is established, the route for all subsequent packets is fixed.

Each packet contains a virtual circuit identifier as well as data. The switches need not make any decisions in routing. The virtual circuit route is programmed into applicable switches for the duration of the connection. The virtual circuit facilities may have the capability of offering "sequencing," "error control" and "flow control." The word "may" is emphasized here because not all virtual circuit facilities provide these services completely

reliably [Ref. 8: pp. 32, 33]. "Sequencing" refers to the fact that since the packets all travel on the same route, they all arrive in the sequence that they were transmitted. "Error control" is a service that ensures that packets not only arrive in sequence but that they arrive correctly. "Flow control" is a mechanism to ensure that the sender does not overwhelm the receiver with data. If the San Francisco host perceives that it cannot process incoming data fast enough and is in danger of running out of buffer space, then it can send a control packet back to the Boston host requesting that it suspend transmission. Long messages are handled better by virtual circuit service. The reason for this is that there is a constant flow of data, and the switches do not have the processing overhead of deciding where to route each packet. It has already been programmed into the switches. A clear disadvantage of virtual circuit switching is that there is no redundancy. If one of the switches in the virtual circuit route becomes congested or goes out of commission, the entire connection is lost. For this reason, datagram service is employed by DDN for reasons of redundancy and graceful degradation.

C. ISO (INTERNATIONAL STANDARDS ORGANIZATION) OSI (OPEN SYSTEMS INTERCONNECTION) MODEL

It has become clear that the one time special purpose approach to communications software development is prohibitively expensive. One solution is to ensure that all hardware is procured from a single vendor who can deliver some assurance of compatibility. Different vendors use different data formats and data exchange conventions. The software development effort becomes a nightmare when dealing with heterogeneous vendors and even between different model lines of a single vendor.

Hence the pressure for developing industry standards for data communications and networking. The benefits of such standards are twofold. If standards become commonly accepted and implemented, vendors would be motivated to implement such standards in their product line. Failure to do so would risk loss of business to competition that did implement standards. Customers would be in a position to demand the standards for equipment that they intend to buy, greatly simplifying their procurement, hardware implementation, and software development problems. One standard to cover this broad field is insufficient. As a result, in 1977 the ISO (International Standards Organization) established a subcommittee that developed an OSI (Open Systems Interconnection) model. This model establishes 7 layered standards for linking heterogeneous computers together. These layers must coexist as mirror images in two computers that communicate with each other. Each of the 7 layers in one host communicates with its mirrored or "peer" layer in the second host. This peer communication is ruled by a set of conventions known as "protocol." Key elements of protocol include "syntax," "semantics" and "timing." Syntax refers to such things as format of data and levels of signaling. Semantics refers to control information for error handling. Timing refers to synchronization of speed and sequence control.

These 7 layers are hierarchical in nature. If host A invokes the layer 7 protocol to transmit information on a process occurring between host A and host B, it is necessary for layer 7 at host A to communicate with its peer layer 7 resident at host B. In order to effect this communication, the services of every layer beneath layer 7 are required in a downward cascading fashion. Layer 7 in host A requests the services of layer 6, which in turn requests the services of layer 5, ...and so on down to layer 2 requesting the

services of layer 1. The physical connection is effected between host A and host B through layer 1. The process of peer level communication then cascades upward as information transmitted to layer 1 in host B feeds information up to layer 7. These 7 ISO OSI layers are discussed below. Stallings does an excellent job discussing these protocols, and the following discussion draws heavily from his book [Ref. 8: pp. 36 thru 50].

1. Physical Layer

This layer covers the conventions of physical interfacing and the rules by which bits of data are passed between two machines. By physical interface we refer to the pin connections, electrical voltage levels, and signal formats [Ref. 4: p. 109]. This layer has 4 important characteristics [Ref. 39]: "electrical," "mechanical," "functional," and "procedural." The most common standard associated with layer 1 is the RS-232-C standard which specifies a standard 25 pin connector among other things. Also associated with the physical layer are the RS-449 standard, and the CCITT X.21 standard.

2. Data Link Layer

Layer 2 is concerned with making reliable the raw bit stream service and physical link specified by layer 1. It provides the means for activating, maintaining, and deactivating the link. It concerns sending blocks of data called "frames" with a checksum for error detection and enforcing frame acknowledgement. These bit oriented protocols are intended to provide:

- Code independent operation and transparency.
- Adaptability to various applications, configurations, and uses in a consistent manner such as point to point, multidrop, and loop configuration.
- Both two way alternate (half-duplex) and two way simultaneous (full-duplex) data transfer.

- High Efficiency.
- High Reliability.

3. Network Layer

The concept of a protocol begins to become hazy at the network layer. It is designed to facilitate communications between systems through a data communications network. It is supposed to provide transparent transmission of data between two transport entities. At a bare minimum, the network layer is to relieve the transport layer above it of the need to know anything about the underlying communications medium that it is using. The network layer is most commonly used to handle the details of a packet switched network. A special case of the network layer is the CCITT X.25 protocol which will be discussed shortly. In a robust implementation, the network level could provide the capability for two devices that are not even connected together on the same network to communicate with each other through one or more intermediate networks. The network layer provides the capability for datagram or virtual circuit transmission of packets. It is responsible for routing and congestion control.

4. Transport Layer

The transport layer has responsibility for ensuring that sequences of data are delivered error free, in proper sequence, and with no losses or duplication of data. If a reliable layer 3 (network layer) is implemented that delivers reliable virtual circuit transmission, then a minimal implementation of the transport layer is required. If layer 3 supports datagrams, or is unreliable, then a robust implementation of the transport layer is required. The transport layer must provide extensive error detection, recovery procedures, and flow control if the underlying

network layer does not provide any one of these functions. It should be noted that layers 4 through 7 are generally referred to as the higher layers.

5. Session Layer

The session layer provides the mechanism for controlling dialogue between two presentation (layer 6) entities. It provides a means for the two layer 6 entities to establish and use a connection. This connection is called a session. A session layer may be two-way simultaneous (full duplex), two-way alternate (half duplex), or one-way (simplex). The presentation layer may require that data not be transmitted until a certain quantum of data (quarantine unit) has accumulated. The session layer provides that capability of blocking the data for the presentation layer. It also provides a checkpointing mechanism, so that if a failure occurs between two checkpoints, the session layer is capable of recovering the dialog between two presentation entities through retransmission of all data since the last checkpoint.

6. Presentation Layer

The presentation layer performs transformations on data in order to provide a standardized application interface and common communications services. It provides code and character set translation and the reformatting of data. It allows for the initial selection and subsequent modifications of transformations used. Typical examples of presentation layer protocols are text compression, encryption, and virtual terminal protocol.

7. Application Layer

The application layer consists of those programs which are designed to be run in a distributed environment.

Typical applications layer programs would be utility programs that support distributed processing such as electronic mail, a transaction server program, file transfer protocol, and a job control language protocol. These programs would in turn be implemented by different applications for execution in a distributed environment.

D. CCITT (CONSULTATIVE COMMITTEE FOR INTERNATIONAL TELEPHONE AND TELEGRAPH) STANDARDS

Through strong leadership and participation by Great Britain, France, Canada, and the United States, two important standards for packet switching have been developed by the CCITT (Consultative Committee for International Telephone and Telegraph). These two standards are the X.25 standard for network user interface and the X.75 standard for network to network interface.

1. Definitions

a. DTE (Data Terminal Equipment)

A DTE is a device, usually belonging to a data communications user, which provides functional and electrical interface to the communications medium. Typical examples are a teleprinter, CRT (Cathode Ray Tube) terminal, or a computer [Ref. 4: p. 357].

b. DCE (Data Circuit-Terminating Equipment)

DCEs are network owned devices that provide an attachment point for users into the network [Ref. 8: p. 352].

c. Gateway

A gateway is a device that connects two network systems, especially if the systems use differing protocols.

A gateway is needed to connect two local area networks, or to connect a local area network into a long haul network [Ref. 8: p. 353].

d. Catanet

A catanet is a collection of packet switched networks that are connected together by gateways [Ref. 8: p. 352].

2. X.25 Standard

The X.25 standard is a special case of the ISO OSI protocol layers 1 through 3. More specifically, it is a standard for interface between a DTE and a DCE. Rosner does an excellent job describing the X.25 standard. The following discussion draws heavily from his work [Ref. 4: pp. 123 thru 139]. X.25 makes extensive use of existing standards. Because virtual circuits are set up, and packets are transmitted between two DTEs, it also has aspects of a DTE to DTE protocol. However, X.25 is generally thought of as a protocol with local significance providing the DTE and DCE interface. In general, X.25 does not guarantee reliable end-to-end (DTE to DTE) transmission, nor does it guarantee flow control. Depending upon implementation however, it can and does provide those services. The X.25 standard was originally ratified in 1976 and has been continuously upgraded over the next several years. In the original and most prevalent implementation, X.25 provides virtual circuit switching.

a. Level 1, Physical Layer

At this layer, the CCITT X.26 or equivalent American EIA (Electronic Industries Association) RS-423 standards define the interface for electrically unbalanced connections. The CCITT X.27 and equivalent EIA RS-422

standards define the interface for electrically balanced connections. Level 1 of the X.25 provides for synchronization between the DTE and DCE.

b. Level 2, Data Link Layer

At this layer, X.25 specifies existing standards for link control procedures. The users are presumed to be interfacing with the network using a relatively high speed, synchronous data link control. Long standing standards in this area exist. These are the ISO HDLC (High Level Data Link Control) protocol and the ANSI (American National Standards Institute) ADCCP (Advanced Data Communications Control Procedure). These protocols involve the use of bit synchronous transmission of discrete blocks of data. A standard level 2 feature of X.25 is the error control mechanism. Errors in a bit stream are detected and corrected by an algorithm called CRC (Cyclic Redundancy Check) for error detection and block retransmission. The CRC algorithm takes the bits transmitted in the "frame address," "control," and "information" fields and divides this long binary number by another standard binary number. The result of course leaves you with a quotient and a remainder. The remainder is the critical number that is the CRC for that packet. This CRC is computed before transmission of the packet and is included in the CRC field. At the packet destination, the CRC is computed again and compared with the CRC value transmitted by the sender. If they differ, then an error is detected and retransmission procedures are initiated. Level 2 of the X.25 provides for error detection, correction through retransmission, and transparent connection between the DTE and DCE.

c. Layer 3, Network Layer

X.25 networks must insure high probability of successful transmission. This generally requires alternate routing and congestion control procedures. The network side of level 3 must provide for delivery of packets in proper sequence and accounting for delivery of packets. This level provides for switching functions permitting multiple connections between different combinations of network ports.

The most popular implementation of X.25 is the virtual circuit implementation alluded to earlier. X.25 has been expanded recently to encompass standards for datagram transmission methodology. However, datagram implementation has yet to gain acceptance for commercial use.

The permanent virtual circuit implementation guarantees connection on demand between a fixed pair of network endpoints. Since all data travels the same circuit between the same two endpoints, there is no need to specify the destination of a packet. This type of implementation though not flexible, is efficient since there is no call initiation and connection delay. The two major X.25 packet structures are briefly illustrated below.

The Call Request packet fields are formatted as follows:

- Flag (8 bits): 01111110, beginning of packet
- Link Address (8 bits): local DTE - DCE connection address
- Link Control (8 bits): used for DTE - DCE error control
- Format Identifier (4 bits): defines type of packet to follow
- Logical Channel Identifier (12 bits): connection identification number
- Packet Type (8 bits): defines function and content of this packet
- Calling Address Length (4 bits): length in digits of sending address

- Called Address Length (4 bits): length in digits of receiving address
- Called Address (up to 60 bits)
- Calling Address (up to 60 bits)
- Facilities Length (6 bits): length in 8 bit bytes of following field
- Facilities Field (up to 512 bits): specifies use of optional network facilities
- Protocol Identifier (32 bits): specifies user level protocols
- User Data (up to 96 bits): user data, such as password
- CRC (Cyclic Redundancy Check): error checking hashing algorithm
- Flag (8 bits): 01111110, end of packet

The Data Transfer packet fields are formatted as follows:

- Flag (8 bits): 01111110, beginning of packet
- Link Address (8 bits): local DTE - DCE connection address
- Link Control (8 bits): used for DTE - DCE error control
- Format Identifier (4 bits): indicates a data packet
- Logical Channel Identifier (12 bits): connection identification number
- Send Packet Sequence Number (3 or 7 bits): sequential number assigned to this packet
- Receive Packet Sequence Number (7 bits): sequential number acknowledging the last successfully received packet from the other user
- More Data Bit (1 bit): 0 if not prepared to receive more data, 1 if clear to receive more data
- User Data Field (up to 1024 bits): actual data, up to 128 characters or bytes
- CRC (Cyclic Redundancy Check): error checking hashing algorithm
- Flag (8 bits): 01111110, end of packet

The two packet structures cited above are the basic architecture upon which all of the other varieties of packets in X.25 are generated. Such packets include call clear, call reset, connection confirmed, packet acknowledgement, as well as others. As stated earlier,

almost all commercial X.25 implementations are virtual circuit connections. In order to assure proper sequence of delivery, the transmitting user must insert the sequence number into each packet.

Additional standards designated X.28 and X.29 have been developed to provide point to point connection transparency for nonintelligent or dumb terminals through a packet switched network. Such interface features are implemented by the network switching centers which must perform PAD (packet assembly disassembly) functions for the dumb terminals.

In the virtual circuit implementation of X.25, flow control is accomplished through the use of a windowing technique. In X.25, the window flow control matches the rate of data transmitted by the sender into the network to the rate of data delivered to the receiver by the network. The window defines the maximum allowable number of packets in transit through the network in one direction for a given connection. It is linked and computed through the use of the current "Send Packet Sequence Number" and the last acknowledged packet. The concept of windowing is explained in greater detail in the section on TCP (Transmission Control Protocol) in Appendix B.

Flow control is implemented between two users, or DTEs, connected through their respective DCE connections. However, it is also necessary to transmit signals such as interrupt packets, reset requests, restart requests, and status inquiries into the network. These are called "Out of Band" signals. Since the timing of these signals is critical to the connection in progress, they are not subject to flow control constraints that apply to other normal packets.

In conclusion, it must be emphasized that the X.25 protocol is strictly a DTE - DCE network interface

protocol. It does not provide for any of the higher level protocols. The X.25 does offer a solution for implementation of higher level user-to-user protocols. It does this by facilitating the layering of the user-to-user higher level protocols on top of the 3 layers implemented by X.25. This makes the higher levels independent of the interfacing of the network.

3. X.75 Standard

The X.75 standard is a supplement to the X.25. CCITT developed it for use between public X.25 networks. It is not likely to be used or even allowed as an interface between public and private networks. It does allow interconnection of a collection of X.25 networks in catanet fashion. The following discussion of the X.75 standard draws heavily from Stallings' work [Ref. 8: pp. 302, 303, 311 thru 314]. The X.75 method of internetworking is a network by network DCE architecture. It specifies a protocol for the exchange of packets between networks that allow a series of X.25 intranetwork virtual circuits to be strung together. X.75 uses the DCE as a gateway, and the gateways must maintain state information about all virtual circuits passing through them. The routing of course is fixed, since virtual circuits are implemented, and all interconnected networks must be X.25.

X.75 specifies STE (Signal Terminating Equipment), which act as DCE level gateways connecting two X.25 networks. The interconnection of X.25 networks via X.75 provides for DTE - DTE virtual circuit switching. This is accomplished through the concatenation of a series of virtual circuits. The system is illustrated by Figure A.1 with one STE serving as a direct gateway between two X.25 networks, or more than one STE serving as gateways between intermediate X.25 networks also.

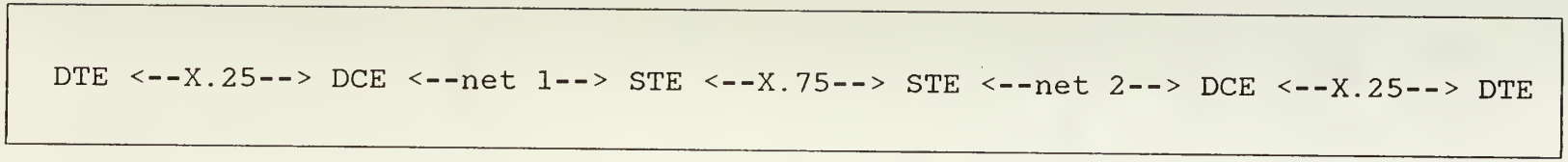

Figure A.1 Interconnection of X.25 Networks via X.75.

Each network section, depicted above as "net 1" and "net 2" are distinct entities with separate virtual circuits, flow control, and error control. As far as the DTEs are concerned, the entire system is transparent and appears as an enlarged X.25 network, rather than an interconnection of different networks. There is no encapsulation of data by the STEs. The same layer 3 header formats used by X.25 are reused. There is no end-to-end protocol. All information has local significance only, just as with X.25 protocol. Because of the 12 bit "Logical Channel Identifier" field in the X.25 header, the maximum number of connections that an STE - STE internet link can handle is 4096. The X.75 control packet format differs from X.25 only in the addition of a "Network Utilities Field," which is used to set up an STE - STE virtual circuit connection.

APPENDIX B

ARPANET AND DDN PACKET SWITCHING INTERNETWORKING METHODOLOGY

The DDN protocol suite provides an interoperable set of subscriber services. In the research community, all subscribers have already implemented the complete protocol suite. The DDN subscriber community is growing rather quickly as existing and new data communications applications are being added to the system. At this time, not all of these applications in the military community have implemented the DDN protocol suite. However, it is incumbent on all subscribers to implement the full DDN protocol suite in a timely manner. The Internet Protocol and Transmission Control Protocol are the foundation of all internetwork data communications through the DDN [Ref. 14: p. 3]. Together, these protocols deliver highly reliable end-to-end datagram internetwork communications.

A. IP (INTERNET PROTOCOL)

1. Background

The Internet Protocol is designed for use in interconnected systems of packet switched data communications networks, otherwise known as catanets. It provides for transmission of long blocks of data in the form of datagrams. The sources and destinations are hosts identified by fixed length addresses. The IP provides for addressing, and if necessary, fragmentation and reassembly of long datagrams for transmission through "small packet" networks. "Small packet" refers to the fact that maximum allowable packet size in a network is severely restricted, thus requiring larger packets that could pass through other

networks to be split one or more times into smaller packets that can pass through the "small packet" network. The following discussion of the IP draws heavily from the Internet Protocol specification [Ref. 32].

The IP is specifically limited in scope. There are no provisions to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host to host protocols. This protocol is called on by host to host protocols in an internetwork environment. IP in turn calls on local network protocols to transmit the internet datagram to the next gateway, or destination host.

2. Operation

The IP performs two basic functions, addressing and fragmentation. Internet modules use the addresses carried in the internet header to transmit datagrams toward their final destination. The internet modules use fields in the internet header to fragment and reassemble internet datagrams when necessary for transmission through small packet networks. The internet module resides in each host engaged in internetwork communications and each gateway that connects two or more networks. These modules share common protocols for interpreting address fields, fragmentation of packets, and reassembly of packets. These modules, especially in gateways, execute procedures for making routing decisions as well as other functions.

Each datagram is treated as an independent entity totally unrelated to any other datagram in the network. There is no virtual circuit. Nor does the internet protocol provide a reliable communication facility. There are no acknowledgements either end-to-end, or hop by hop. There is no error control for the data, only a header checksum. There is no flow control, and there are no retransmissions. Errors detected are reported through the ICMP (Internet

Control Message Protocol) [Ref. 33], which is implemented as part of the IP module. The IP uses four key mechanisms to provide its service. These are described below.

a. Type of Service

The type of service mechanism uses a generalized set of parameters that define service choices offered by the networks that make up the internet. It is used by gateways to determine what set of parameters to invoke for a particular network and what network to select for the next hop, or the next gateway when routing an internet datagram.

b. Time to Live

This is a parameter that specifies the maximum amount of time that an internet datagram is allowed to exist within the internet. It is set by the sender and is decremented by every switch that it passes through. If the time to live reaches zero, the datagram is destroyed.

c. Options

Options provide for control functions useful or needed in certain circumstances, but not necessary in most common communications. They include provisions for timestamps, security, and special routing.

d. Header Checksum

This provides verification that the information in the header that is used in processing the internet datagram has been transmitted correctly. The data may contain errors, but this checksum will not detect it. If the entity processing the datagram computes a checksum that is not in agreement with the transmitted header checksum, the datagram is discarded immediately.

3. Relationship With Other Protocols

Figure B.1 illustrates the place of the IP in relation to the protocol hierarchy and scheme for host and terminal interoperability. The higher level protocol TCP (Transmission Control Protocol) interfaces with IP from above and is layered on top of IP. IP is layered on top of one of the two network access protocols, either the ARPANET 1822 Access Protocol, or the DDN Standard X.25 protocol. The DDN Standard X.25 currently under development is an adaptation of the CCITT X.25 Standard. It is expected to be implemented in the near future. In the interim, DDN Basic X.25 offers interoperability with other users using DDN Basic X.25. Eventually DDN Basic X.25 service is to be phased out some time after DDN Standard X.25 is implemented.

4. Model of Operation

The following scenario illustrates the transmission of a datagram from one application process to another. In this scenario, illustrated by Figure B.2, we will go through two networks, and cross one intermediate gateway. The sending application program prepares its data and calls on its own Internet Module to prepare for transmission of a datagram. The Sending Host Application passes the "destination address" and other parameters as part of the call. The Internet Module resident in the Sending Host builds the datagram header and attaches the data to it. The Internet Module determines a local network destination address for this internet, Local Network 1. In this case, it is the address of a gateway. It sends this datagram and the local network address to the Local Network Interface (LNI-1). The Local Network Interface (LNI-1) serving the Sending Host, creates a local network header, and attaches the datagram to it. It then transmits this datagram via Local Network 1.

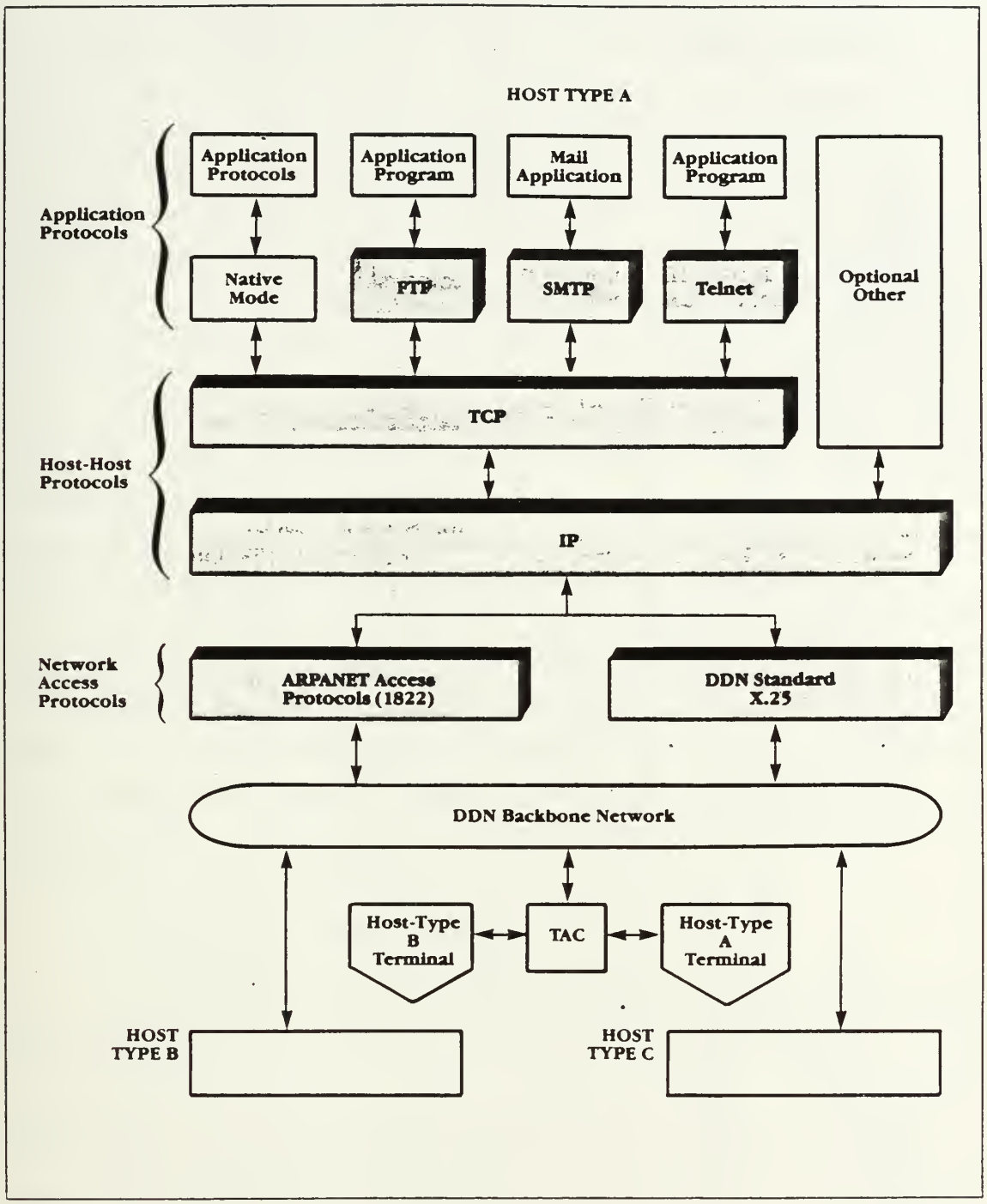

Figure B.1 DDN Protocol Hierarchy, Host and Terminal Interoperability.

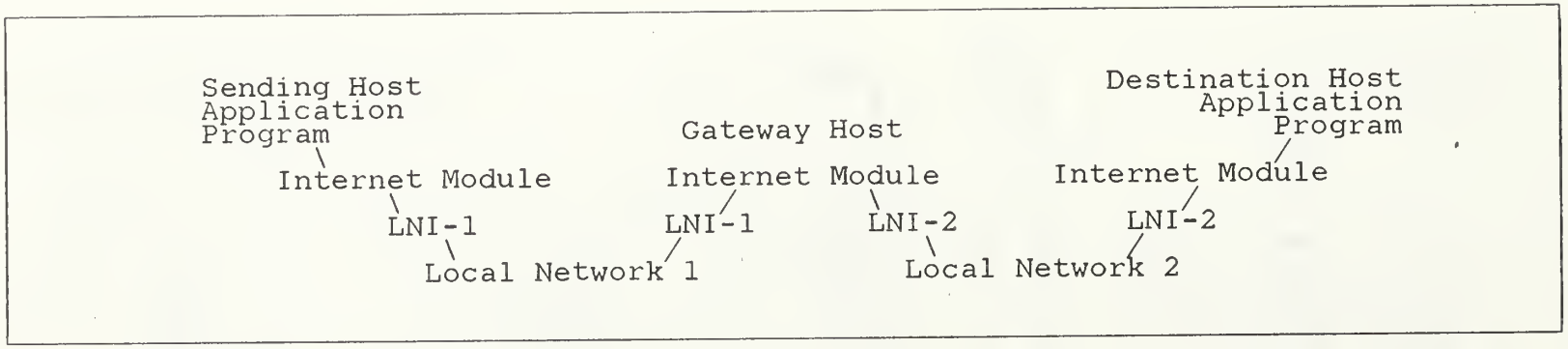

Figure B.2 IP Datagram Transmission Path.

The datagram arrives at a Gateway Host wrapped in the local network header. The Local Network Interface (LNI-1) for Local Network 1, strips off the datagram header and turns it over to the Internet Module resident in the Gateway Host. The Internet Module determines from the internet address, that the datagram is to be forwarded to another host in a second network. The Internet Module then determines a Local Network 2 address for the Destination Host. It calls the Local Network Interface (LNI-2) for Local Network 2 to send the datagram to the Destination Host. This Local Network Interface (LNI-2) builds a Local Network 2 header, attaches the datagram to this header, and transmits the datagram through Local Network 2 to the Destination Host. At the Destination Host, the Local Network Interface (LNI-2) strips off the Local Network 2 header and passes it to the Internet Module resident at the Destination Host. The Internet Module determines that the datagram is for an application program in the Destination Host. It strips the datagram header, and passes the data to the appropriate applications program in response to a systems call. It also passes the source address and other parameters during execution of the call.

5. Functional Description

The purpose of the IP is to move datagrams through an interconnected set of networks. Internet modules reside in hosts and gateways in the internet system. Datagrams are transmitted from one internet module to another through individual networks based on interpretation of the internet address. The internet address is an important mechanism in the IP. When datagrams are routed from one internet module to another, they may need to traverse a network whose maximum packet size is less than the size of the datagram. The IP implements a fragmentation mechanism to solve this problem.

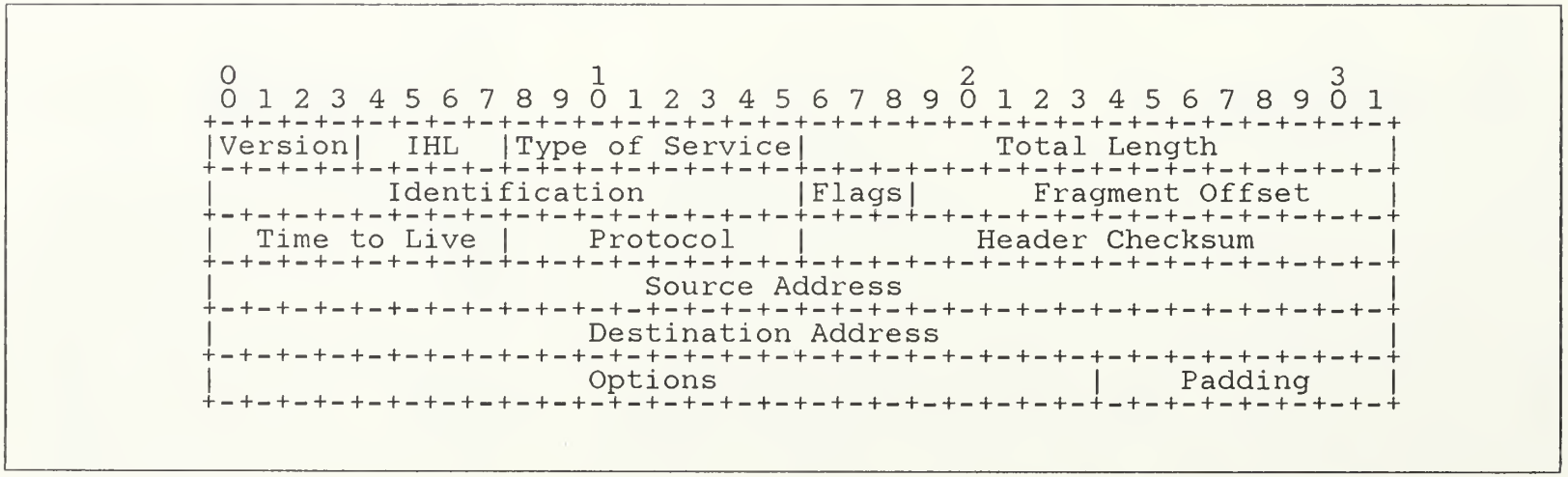

Figure B.3 Example of an Internet Datagram Header.

a. Internet Header Format Specifications

A standard internet header is illustrated in Figure B.3 and a brief description of the fields follows.

- Version (4 bits): Describes the format of the internet header. Version 4 is standard.
- IHL (Internet Header Length) (4 bits): Length of the internet header in 32 bit words. The minimum internet header length is 5 (20 bytes).
- Type of Service (8 bits): This field provides an indication of the abstract parameters specifying quality of service desired. The parameters that can be specified involve precedence, delay, throughput, and reliability.
- Total Length (16 bits): The maximum datagram size is 65,535 bytes. Such long datagrams are impractical for most hosts and networks. All hosts must be prepared to accept datagrams of up to 576 bytes. 576 bytes is the recommended default value for transmission of datagrams.
- Identification (16 bits): A unique identifying value assigned by the sender to aid in reassembling fragments of a datagram.
- Flags (3 bits): Various control flags. Bit 0 is reserved. Bit 1 allows/forbids fragmentation. If bit 1 is set to a default of zero, fragmentation is permitted, setting it to one forbids fragmentation. Bit 2 indicates more/last fragment. A zero indicates the last fragment. A one indicates more fragments.
- Fragment Offset (13 bits): This field indicates where in the original datagram a fragment belongs. The fragment offset is measured in units of 8 bytes (64 bits).
- Time to Live (8 bits): This field indicates the maximum length of time a datagram is allowed to exist in the internet system from the moment of transmission. This field is decremented by 1 bit every time it traverses a module that handles it. This field is modified in internet header processing. The intent of this field is to destroy datagrams that cannot be delivered and to give an upper bound to the maximum datagram lifetime.
- Protocol (8 bits): Indicates the next level protocol used in the data portion of the internet datagram.
- Header Checksum (16 bits): A checksum on the header only, recomputed and verified at each point the internet header is processed.
- Source Address (32 bits)
- Destination Address (32 bits)
- Options (variable length): Options may or may not appear in datagrams. They must be implemented by all IP modules (host and gateway). This field covers a wide variety of parameters such as security, handling

restrictions, transmission restrictions, recording of routing, specification of routing (this is equivalent to setting up a virtual circuit), stream identification for SATNET (Satellite Network), and internet timestamps. In normal unclassified communications, options are not specified.

b. Addressing

A distinction should be made between names, addresses and routes. A name indicates what we seek. An address indicates where it is located. A route indicates how to get to that address. The IP is primarily concerned with addresses. Higher level (layer) protocols are responsible for mapping out a correspondence between names and addresses. The IP maps internet addresses to local network addresses. It is the task of lower level protocols within the local network or gateway, to map a correspondence between local network addresses and routes. Addresses are 32 bits or 4 octets (bytes) in length. It begins with a network number, followed by the local address. There are three options for addressing schemes which cover the gamut from a large number of local networks with a small number of hosts, to a small number of local networks with a large number of hosts.

c. Fragmentation

It is necessary to fragment an internet datagram when it originates in a network that allows a large packet size and on the way to its destination must traverse a network with a smaller maximum packet size. An internet datagram can be marked "don't fragment." If it is marked this way, it will never be fragmented. However, if the datagram so marked cannot be delivered to its destination without fragmenting it, the datagram will be discarded.

The IP fragmentation and reassembly procedure needs to be able to fragment datagrams into almost any

number of pieces that can later be reassembled. The receiver of fragments uses the datagram header "Identification" field to ensure that fragments of differing datagrams are not mixed. The "Fragment Offset" field tells the receiver what position the fragment held in the original unfragmented datagram. The "Fragment Offset" and "Total Length" fields determine the portion of the original datagram covered by this fragment. Finally, the "More Fragments" flag, when reset, indicates the last fragment. Together, the information from all of these fields is sufficient to reassemble a fragmented datagram.

The "Identification" field is used to uniquely identify datagrams for a given connection between two hosts. This field must be assigned a value by the sending host when the datagram is originated. It must uniquely identify that datagram for the maximum allowable life of the datagram on the internet. The "Identification," "Source Address" and "Destination Address" fields uniquely identify every datagram on the internet from all others. In addition, it is the responsibility of the originating host to set the "More Fragments" flag and "Fragment Offset" field to zero.

Suppose that a long datagram reaches a gateway, and the gateway determines that the datagram must be fragmented in order to reach its destination. The gateway will take the data portion of the original datagram and split it in at least two parts. For the purpose of this discussion, let us assume that it will be split into two parts. The first portion must end on an 8 byte, or 64 bit, boundary. In other words, the size in bytes of the first portion must be divisible by 8. In fact, if there are more than two portions, or fragments, the length in bytes of each successive fragment must also be evenly divisible by 8. The last fragment is an exception. The last fragment need not end on an 8 byte boundary, but it must end on a byte

boundary. The number of 8 byte blocks of data in the first fragment are referred to as NFB (Number of Fragment Blocks). The fields that made up the internet header of the original datagram are copied into the internet header fields of the two new datagrams. The first fragment is placed in the first new internet datagram. The "Total Length" field of the internet header is is set to the length of the first new datagram and the "More Fragments" flag is set to 1. The second fragment is placed in the second new internet datagram, and the "Total Length" field of internet header is set to the length of the second new datagram. In the second new datagram, the internet header "More Fragments" flag is equal to zero, the value in the original datagram internet header. The "Fragment Offset" field of the second new datagram will contain the sum of the original internet header "Fragment Offset" plus the NFB (Number of Fragment Blocks) computed earlier for the first fragment. The procedure described above can be generalized for a multiple fragmentation split in much the same fashion as was described above for the two way fragmentation split.

Reassembly of the fragments of an internet datagram at a gateway or destination host requires that the "Identification," "Source," "Destination," and "Protocol" internet header fields of the fragmented datagrams be the same. This is checked by the internet module at the gateway or destination host. Reassembly is accomplished by placing the data portion of each fragment in the relative position indicated by the "Fragment Offset" field of the respective internet headers. The first fragment will have the "Fragment Offset" set to zero. The last fragment can be identified by the "More Fragments" flag being set at zero.

d. GGP (Gateway to Gateway Protocol)

Gateways implement the IP so that datagrams can be forwarded between networks. The GGP is also implemented to coordinate routing and other internet control information [Ref. 40]. In a gateway, the higher level protocols are not required. The GGP as well as the ICMP (Internet Control Message Protocol) are added to the IP module as illustrated in Figure B.4.

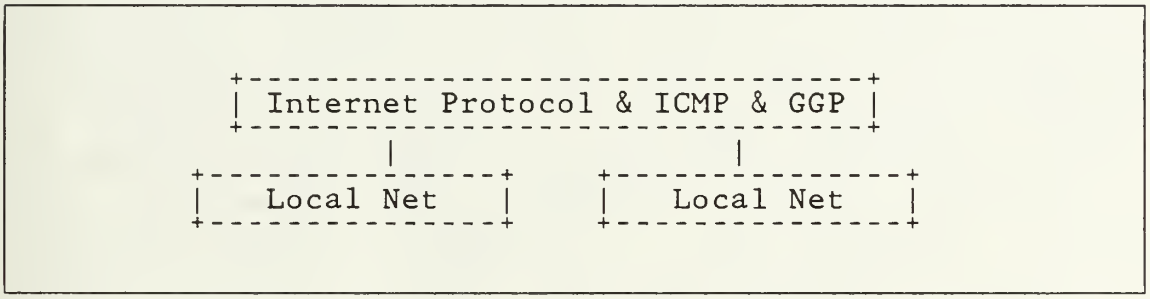

Figure B.4 Gateway Protocols.

6. Implementation Remarks and ICMP (Internet Message Control Protocol)

Implementation of the IP must be conservative in its sending behavior and liberal in its receiving behavior. It must be careful to send well formed datagrams, but must accept any datagram that it can interpret. Such an implementation is geared toward offering robust data communications service. The basic service provides for fragmentation of datagrams at gateways, and reassembly being performed by the destination host's internet module. Fragmentation and reassembly of datagrams within a network or by private agreement between gateways of a network is allowed. Such fragmentation and reassembly is transparent

to IP and the higher level protocols, and is called "network dependent" or "intranet" fragmentation. Internet addresses distinguish sources and destinations to the host level and provide a protocol field as well. It is assumed that each protocol will provide for whatever multiplexing is necessary within a host.

An integral part of the IP is the ICMP (Internet Control Message Protocol) [Ref. 33]. Periodically a gateway or destination host will communicate with a source host to report errors in datagram processing. For such purposes, the ICMP is used. ICMP uses the basic support of IP as if it were a higher level protocol, but in essence, ICMP is an integral part of IP and must be implemented by every IP module. Typically, ICMP messages are sent when a datagram cannot reach its final destination, when a gateway does not have the buffering capacity to forward a datagram, and when the gateway can direct the host to send traffic on a shorter route. The purpose of ICMP is to send control messages about problems in the communications environment, not to make IP reliable. End to end reliability must be provided by higher level protocols if required. To avoid the infinite recursion of messages about messages, no ICMP messages are sent about ICMP messages. When dealing with fragmented datagram problems, ICMP will only transmit messages concerning fragment zero, the last fragment.

B. TCP (TRANSMISSION CONTROL PROTOCOL)

The TCP (Transmission Control Protocol) was developed to provide highly reliable host to host communication in packet switched networks, and interconnected systems of packet switched networks. TCP focuses its attention on military computer communications requirements. It attempts to deliver robust service despite communication unreliability,

and unavailability in the presence of congestion. Many of these problems are found in the civilian and government sector as well. The discussion on TCP that follows draws heavily from DARPA's Internet Program TCP Protocol Specification [Ref. 34].

1. Introduction

TCP is a connection oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multinetwork applications. TCP provides the ability to perform reliable interprocess data communications between two hosts that reside on two separate but interconnected networks. The major assumption is that TCP utilizes simple but potentially unreliable datagram service from lower level protocols. In principal, TCP should be able to function above a wide range of communications systems including hard wired connections, packet switched networks, and circuit switched networks. TCP fits into a layered protocol architecture just above basic IP (Internet Protocol) as illustrated in Figure B.1. IP provides the capability for TCP to send and receive variable length segments of information enclosed in IP datagram "envelopes." The TCP specification describes an interface to the higher level protocols which can be implemented in front end processors, provided that a suitable host to front end processor protocol is implemented. TCP is intended to provide reliable process-to-process data communications service in a multinetwork environment. It is intended to be the basis of host to host protocol in common use throughout the DDN.

2. Interfaces

Referring back to Figure B.1, TCP interfaces from above with user or application processes. From below, TCP interfaces

with IP. The interface between an application process and TCP is conducted through a set of calls, similar to calls made to the operating system for input/output service and file manipulation. Calls to TCP are invoked by applications processes to open and close connections, and to send and receive data on established connections. TCP must be able to asynchronously communicate with application programs. Implementors of TCP are allowed considerable freedom. However, a minimum functionality is required at the TCP/user interface for any valid implementation. The interface between TCP and lower level protocols is basically unspecified. It is assumed that the two levels can pass information to each other asynchronously. Normally, the lower level protocol specifies this interface. TCP is designed to work in a very general environment of interconnected networks.

3. TCP Header Format Specification

TCP segments are sent as IP datagrams in the DDN implementation. The TCP header follows the internet header, supplying information specific to the TCP protocol. Figure B.5 illustrates the basic TCP header format. The various fields in the TCP header are briefly summarized below.

- Source Port (16 bits): The source port number.
- Destination Port (16 bits): The destination port number.
- Sequence Number (32 bits): Sequence number assigned to the first data byte in this segment.
- Acknowledgement Number (32 bits): The value of the next sequence number that the sender of this segment is expecting to receive.
- Data Offset (4 bits): The number of 32 bit words in the TCP header. This indicates where the header ends and the data begins.
- Reserved (6 bits): Reserved for future use.

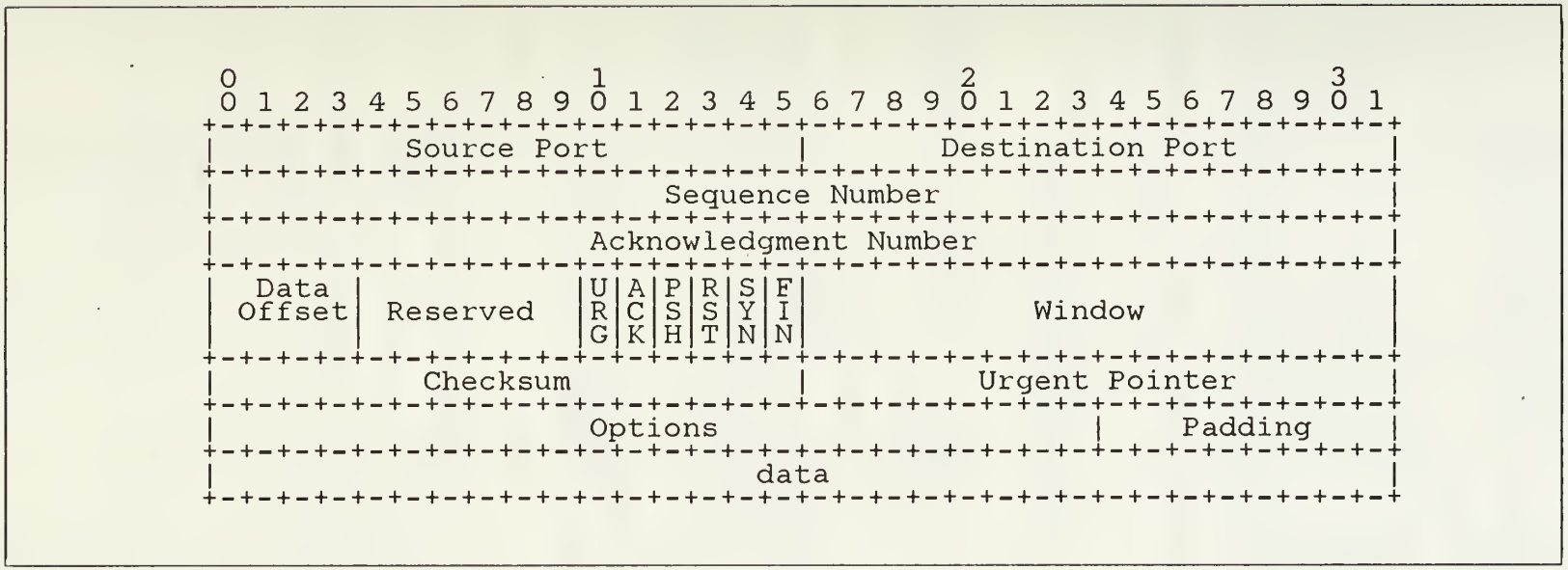

Figure B.5 TCP Header Format.

- Control Bits (6 bits): From left to right, defined below.
  - URG (1 bit): Urgent Pointer field significant.
  - ACK (1 bit): Acknowledgment field significant.
  - PSH (1 bit): Push function.
  - RST (1 bit): Reset the connection.
  - SYN (1 bit): Synchronize sequence numbers.
  - FIN (1 bit): No more data from sender.
- Window (16 bits): The number of bytes beginning with the sequence number in the "Acknowledge" field that the sender of this segment is willing to accept.
- Checksum (16 bits): Checksum figure computed through one's complement arithmetic on all 16 bit segments of header and text.
- Urgent Pointer (16 bits): The current value of the urgent pointer computed as a positive offset from the sequence number in this segment. Points to the sequence number of the byte of data immediately following the urgent data.
- Options (variable): Occupy space at the end of the TCP header and are a multiple of 8 bits in length. There are two cases for the options format.
  - Case 1: A single byte of an option kind.
  - Case 2: A byte of option kind, a byte of option length, and the actual option data bytes. The only meaningful option implemented to date is the 16 bit "Maximum Segment Size" which specifies the maximum receivable segment size that the sender of the option is willing to accept. This option is set once at the beginning of a connection.
- Padding (variable): TCP header padding is used to ensure that the TCP header ends, and data begins on a 32 bit boundary. Padding is composed of zeros.

4. Operational Overview

As stated earlier, TCP is supposed to provide reliable, secure logical circuit or connection service between a pair of processes. Providing this service on top of the less reliable IP requires facilities in areas described below.

a. Basic Data Transfer

TCP transmits a continuous stream of bytes in each direction between two user processes by packaging some number of bytes into segment blocks for transmission through the internet system. The TCPs decide when to block and forward data at their own convenience. In some situations, the application program may want to explicitly transmit data at a given point, rather than letting TCP decide when to transmit data. A "push" function is provided for this purpose. The "push" causes TCP to immediately transmit all data in the transmission queue.

b. Reliability

TCP must be able to recover from data that is damaged, duplicated, delivered out of order, or is lost by the internet communications system. This is accomplished through the assignment of a "Sequence Number" to each byte that is transmitted. It requires a positive acknowledgement (ACK) from the receiving TCP. If an acknowledgement is not received within a specified timeout interval, the data is retransmitted. The receiving TCP uses the sequence numbers to correctly order segments that may be received out of order, and to eliminate any duplicate transmissions. Damaged data in the form of garbled or erroneous transmission is handled by adding a checksum to each segment transmitted. This checksum is recomputed by the receiving TCP. If it is not in agreement with the transmitted checksum, the segment in question is discarded. As long as TCP continues to function and the internet does not become completely severed, no transmission errors will affect the correct delivery of data. TCP recovers from internet communication system errors.

c. Flow Control

A receiving TCP has the ability to control the flow of data transmitted by the sending TCP. The receiving TCP does this by returning a "Window" with every acknowledgement (ACK) of segments received back to the sending TCP. The "Window" indicates an acceptable range of segment numbers that the sending TCP may transmit beyond the segment number currently acknowledged. In other words, it indicates an allowed number of bytes that the sending TCP may transmit before receiving further permission. In this manner, the receiving TCP can ensure that the buffer space allocated to the current connection does not become overloaded, or that data is not received faster than the receiving TCP can process it. This is an essential mechanism for asynchronously connected processes.

d. Multiplexing

TCP provides a set of addresses or ports within each host to allow several processes to conduct communications with a single host simultaneously. The port address, network address, and local host address are concatenated to form a socket. A pair of sockets uniquely identifies each connection. A socket may be simultaneously used with a number of connections. The binding of ports to processes is handled independently by each host. It is useful to permanently attach frequently used service oriented processes to fixed sockets and make these known to the public that uses them.

e. Connections

The reliability and flow control mechanisms require information which includes socket identifier, sequence numbers, and window sizes to define a connection.

Each connection is identified by a pair of sockets which identifies its two sides. When two processes wish to communicate, their TCPs must establish a connection. This requires the initialization of the status information just described. When their communication is completed, the connection is terminated and the resources used by the connection are freed. Connections must be established between unreliable hosts over the unreliable internet. A handshake mechanism with clock based sequence numbers is used to avoid erroneous initialization of connections.

f. Precedence and Security

TCP users may specify the security and precedence of their communication. Default values are assigned when this is not specified.

5. Details of Implementation Philosophy

The internetwork environment consists of hosts connected together by a network. These networks may be local area networks such as ETHERNET, or wide area networks such as ARPANET, but in any case they are based on packet switching technology. In the internetwork environment, these networks are connected by gateways. Processes are viewed as active elements of a host computer. Terminals, files, and other input/output devices communicate with each other through the use of processes. Thus all communication is viewed as interprocess communication.

a. Model of Operation

Processes communicate with each other by calling on the TCP module and passing buffers of data as arguments. TCP wraps up this data into segments and in turn calls on the internet module to transmit each segment to the destination host's TCP module. The receiving TCP module

unwraps the segment received, and passes the data to the receiving process through a buffer and notifies the receiving process. TCPs include control information in the segments transmitted to ensure reliable ordered transmission of data.

TCP uses the internet module to provide the interface to the local network. The internet module wraps TCP segments inside internet datagrams and routes these datagrams to the destination internet module or an intermediate gateway. To transmit the data through a local network, it is yet again wrapped up inside a local network packet. Packet switches may perform further packaging (wrapping), fragmentation, or other operations to deliver the local packet to the destination internet module. At a gateway between networks, the internet datagram is unwrapped from its local packet and examined to determine through which network the internet datagram should be routed next. The internet datagram is then wrapped up again in a packet established by the local protocol of the next network it is to travel through. It is routed through to the next gateway, or to the destination host, if it resides in that network. The internet module that serves the destination host reassembles the datagram if it was fragmented. It then unwraps the TCP segment from the datagram and passes it to the destination TCP module.

b. Host Environment

TCP is assumed to be a module in an operating system. Users access TCP in the same manner that they would access the file system. TCP needs to call on other operating system functions, for example, to manage data structures. The actual interface to the network is assumed to be controlled by a device driver module. TCP does not actually interface with the device driver module. TCP

interfaces with the internet module (IP). The internet module actually calls on the device driver module. The functions of TCP allow implementation in a front end processor. However, in such a scheme a host to front end protocol must provide the TCP/user interface.

c. Interfaces

A user process makes calls on the TCP module to OPEN and CLOSE connections, to SEND or RECEIVE data, or to obtain STATUS about a connection. The TCP/IP interface provides for calls to send and receive datagrams addressed to TCP modules resident in hosts anywhere in the internet system. These calls require parameters for passing the "address," "type of service," "precedence," "security," and other protocol information. Figure B.1 illustrates where TCP lies relative to the protocol hierarchy in the DDN. It is expected that TCP be able to support higher level protocols efficiently. It should be easy to interface TCP with higher level DDN protocols such as FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), and TELNET (Network Virtual Terminal Protocol).

d. Reliable Communication

Transmission reliability is accomplished through the use of "Sequence Numbers" and "Acknowledgment Numbers." In concept, each byte of data transmitted is assigned a sequence number. When a segment is transmitted, the first byte of data in the segment is identified by the segment "Sequence Number." Segments also contain an "Acknowledgment Number" which is the sequence number of the next expected byte to be transmitted in the reverse direction. In other words, it is the sequence number of the last byte received plus 1. When TCP transmits a segment containing data, it places a copy of that segment in a buffer called the

retransmission queue and starts a timer. The sending TCP waits for an acknowledgment for the segment transmitted. If it receives an acknowledgment for that segment prior to expiration of the timer, the segment is deleted from the retransmission queue. If the timer expires and no acknowledgment is received, then the segment is retransmitted, and the cycle is repeated with a copy remaining in the retransmission queue. An acknowledgment by a receiving TCP does not guarantee that the data has been delivered to the receiving host user process. It indicates that the receiving TCP module has acknowledged responsibility for doing so.

A flow control mechanism is employed to regulate the flow of data between TCP modules. The receiving TCP module advertises a "Window" to the transmitting TCP module. It should be remembered, that two way communication is often effected simultaneously. So both TCP modules are transmitting and receiving concurrently, and thus both advertise a "Window" to the opposite of the pair. The "Window" specifies the number of bytes, starting with the "Acknowledgment Number" that the receiving TCP is currently prepared to receive.

e. Connection Establishment and Clearing

Unique addresses must be available for each TCP module, since it will likely be serving many processes, and each of these processes needs to be identified. The internet address identifying the TCP module is concatenated with the port identifier to create a "socket" which is unique throughout all the networks connected together. A connection is fully specified end-to-end by a pair of sockets. A local socket can participate in many connections to different foreign sockets. Connections can support transmission in both directions simultaneously (full duplex).

A connection is specified in the OPEN call by the local port and foreign socket arguments. TCP supplies a short local connection name to this connection, a nickname of sorts, by which the user process can refer to the connection in subsequent calls. State information about a connection consists of several items of data. This connection state information is stored in a data structure known as the TCB (Transmission Control Block) which is similar in concept to the PCB (Process Control Block) that an operating system maintains about an executing process in a multiprogramming environment. The OPEN call specifies whether connection establishment should be actively pursued, or passively waited for. A passive OPEN request indicates that a process is willing to accept incoming connection requests, rather than attempting to initiate a connection. Processes requesting passive OPENs may be willing to accept connection with any caller. In this case, a foreign socket of all zeros is used to denote an unspecified socket. Unspecified foreign sockets are allowed only on passive OPEN requests. A service process, such as a "login" procedure, would issue a passive open request with an unspecified foreign socket. Well known sockets are a convenient mechanism for associating a socket address with a standard service. Such would be the case for higher level protocol processes such as TELNET, FTP, SMTP, and RJE (Remote Job Entry).

Processes can issue passive OPENs and wait for matching active OPENs from foreign processes. TCP will inform the local process when the connection has been established. Two processes that issue active OPENs to each other simultaneously will be correctly connected. There are two basic cases for handling for handling a connection between a local passive OPEN call and a foreign active OPEN call. In case 1, the local passive open call has fully

specified the foreign socket, so the match must be exact. In case 2, any foreign socket is acceptable as long as the sockets match. In between lie cases that limit connection to a restricted set of foreign sockets. OPEN calls are recorded in the TCB (Transmission Control Block).

The procedure to establish a connection utilizes the "SYN" (Synchronize) control flag and involves an exchange of three messages known as a "three way handshake." A connection is initiated when a segment containing a "SYN" (Synchronize) flag rendezvous with a waiting TCB (Transmission Control Block) entry created by the local user process OPEN command. A connection is initiated when the local and foreign sockets match. The connection becomes "established" when "Sequence Numbers" have been synchronized in both directions. The clearing of connections also involves the exchange of segments which contain the "FIN" (Finish) control flag.

f. Data Communications

Data that flows on a connection is asynchronous, and therefore can be thought of as a stream of bytes. Data flowing in a given direction is normally stored in a buffer of the receiving TCP module until it is filled, or reaches a high water mark. At this point, the data is normally transferred to the receiving process. The sending TCP has the option of specifying to the receiving TCP that data in the current segment, and all previous segments, should be transferred to the user process immediately. The sending TCP accomplishes this through setting the "PSH" (Push) control flag. This will automatically cause all buffered data in the receiving TCP segment to be transferred to the user process. This buffering process can and does occur at both ends of the sending process. The sending TCP can also wait until an appropriate amount of data is accumulated in

its send buffer before sending segments to the receiving TCP. When the user process indicates that the data should be "pushed," the buffer is cleared and transmitted immediately. Likewise, when the segment containing the "PSH" flag is received, all data in the receive buffer is cleared and transferred to the receiving process.

In addition to the "PSH" control flag, there is an "URG" (urgent) control flag. This indicates to the receiving TCP that there is urgent information in the current segment that should be processed immediately. The "Urgent Pointer" field specifies the "Sequence Number" of the byte immediately following the end of the urgent data.

g. Precedence and Security

Precedence and security is provided for through use of the Internet Protocol "Type of Service" field and "Security" option.

6. TCP Quiet Time Concept

Sequence numbers are 32 bit fields, therefore the possible combinations of different sequence numbers that can be assigned by a sending TCP are 2 to the 32nd power, or 4,294,967,296 in base 10. The reason for such a wide range, is to attempt to limit the possibility of two segments with the same sequence number but different data existing in the internet concurrently. Even if a sending TCP were to continuously transmit data bytes at the rate of 2 megabits/second, it would take approximately 4.5 hours to cycle through all 4,294,967,296 sequence numbers. So the chance of duplicate segment numbers for different data appears highly unlikely. At 100 megabits/second, the cycle time is reduced to 5.4 minutes, which is short, but still exceeds the internet datagram "Time to Live" upper bound of 256. Even if it took a datagram one second to be passed

from one internet module to another internet module, this would still only allow a datagram to exist for 256 seconds, roughly 4.25 minutes, before it would be discarded.

However, if the sending TCP host crashes, it may well present a problem. Unless the sending TCP host remembers the last "Sequence Number" transmitted, it is likely to reset the counter at some arbitrary point such as 0. In this case, there is a distinct possibility of transmitting a segment with a sequence number equal to the sequence number of another segment transmitted before the sending host crashed. Thus two segments of differing data but with the same sequence numbers may coexist in the internet. In the absence of knowledge about the sequence numbers used in a particular connection, the TCP specification recommends the sending host delay for MSL (Maximum Segment Lifetime) seconds before emitting segments on the connection. This will allow time for segments from the earlier connection incarnation to drain from the system. We can normally assume that MSL is equal to the IP "Time to Live" maximum of 256 seconds. If for some reason, it is found that conditions may exist wherein a datagram may take more than one second to travel from one internet module to the next, then the MSL must be adjusted upwards accordingly. Prudent implementation of TCP/IP dictates that a host brought back online deliberately delay emitting segments for one MSL after recovery from a crash. This is called the "Quiet Time" specification. Hosts which prefer to avoid waiting must be willing to risk possible confusion of old and new packets at a given destination. In a supply and logistics transaction environment, failure to implement "Quiet Time" specification could prove disastrous.

APPENDIX C

GLOSSARY OF ACRONYMS

AA:	American Airlines
ACK:	Acknowledgement
ADCCP:	Advanced Data Communications Control Procedure
ADP:	Automatic Data Processing
AIS:	Automated Information System
AMS2:	Aviation Structural Mechanic Second Class
ANSI:	American National Standards Institute
APL:	Allowance Parts List
ASO:	Navy Aviation Supply Office
ATTIS:	American Telephone & Telegraph Information Systems
BBN:	Bolt, Baranek & Newman
BIOT:	British Indian Ocean Territory
BOC:	Bell Operating Company
BSC:	Binary Synchronous Communications
BT3:	Boiler Technician Third Class
C3I:	Command, Control, Communications and Intelligence
CAIMS:	Conventional Ammunition Integrated Management System
CAS:	Contract Administration Service
CASREP:	Casualty Report
CCITT:	Consultative Committee for International Telephone and Telegraph Standards
CNO:	Chief of Naval Operations
COMNAVAIRPAC:	Commander, Naval Air Forces, U. S. Pacific Fleet
COMNAVLOGPAC:	Commander, Naval Logistics Command, U. S. Pacific Fleet
COMNAVTELCOM:	Commander, Naval Telecommunications Command
CONUS:	Contiguous United States (48 States)
COSAL:	Coordinated Shipboard Allowance List

CPU:	Central Processing Unit
CRC:	Cyclic Redundancy Check
CRT:	Cathode Ray Tube
CV:	Aircraft Carrier (conventionally powered)
DAASO:	Defense Automated Addressal Systems Office
DARPA:	Department of Defense Advanced Research Projects Agency
DBMS:	Data Base Management System
DCA:	Defense Communications Agency
DCASMA:	Defense Contract Administration Services Management Area
DCASPRO:	Defense Contract Administration Services Plant Representative Office
DCASR:	Defense Contract Administration Services Region
DCE:	Data Circuit-Terminating Equipment
DDG:	Guided Missile Destroyer
DDN:	Defense Data Network
DDS:	Directory/Dictionary System
DEC:	Digital Equipment Corporation
DLA:	Defense Logistics Agency
DLSC:	Defense Logistics Services Center
DoD:	Department of Defense
DODIIS:	Department of Defense Intelligence Information System
DSP:	Display System Protocol
DTE:	Data Terminal Equipment
DUSD:	Deputy Under Secretary of Defense
EIA:	Electronic Industries Association
FD:	Functional Description
FIPS:	Federal Information Processing Standard
FMSO:	Navy Fleet Material Support Office
FTP:	File Transfer Protocol
FY:	Fiscal Year
GGP:	Gateway to Gateway Protocol
GS:	General Schedule
GSA:	General Services Administration

GTE:	General Telephone & Electronics
HDLC:	High Level Data Link Control
HSLN:	High Speed Local Network
IBM:	International Business Machines
ICMP:	Internet Control Message Protocol
ICP:	Inventory Control Point
ICST:	Institute for Computer Sciences and Technology
IMA:	Intermediate Maintenance Activity
IP:	Internet Protocol
ISO:	International Standards Organization
JCL:	Job Control Language
LNI:	Local Network Interface
MCRL:	Master Cross Reference Listing
MFT:	Multifunction with a Fixed number of Tasks
MINITAC:	smaller version of a TAC (Terminal Access Controller)
ML-N:	Management List - Navy
MOCAS:	Mechanization of Contract Administration Services
MS-DOS:	Microsoft - Disk Operating System
MSL:	Maximum Segment Length
MVS:	Multiple Virtual Systems
NARDAC:	Navy Regional Data Automation Center
NATO:	North Atlantic Treaty Organization
NAVMASSO:	Navy Management Systems Support Office
NAVSEA:	Naval Sea Systems Command
NAVSUP:	Naval Supply Systems Command
NBS:	National Bureau of Standards
NIC:	Network Information Center
NIIN:	National Item Identification Number
NIR:	National Item Record
NIU:	Network Interface Unit
NRC:	National Research Council
NSA:	National Security Agency
NSC:	Naval Supply Center

NSD:	Naval Supply Depot
NSN:	National Stock Number
OS:	Operating System
OSI:	Open Systems Interconnection
PBX:	Public Branch Exchange
PC:	Personal Computer
PC-DOS:	Personal Computer - Disk Operating System (IBM)
PCB:	Process Control Block
PSN:	Public Switched Network
RFC:	Request For Comments
RIC:	Routing Identifier Code
RJE:	Remote Job Entry
SACDIN:	Strategic Air Command Digital Network
SAMMS:	Standard Automated Material Management System
SATNET:	Satellite Network
SDLC:	Synchronous Data Link Control
SK1:	Storekeeper First Class
SKC:	Storekeeper Chief
SMTP:	Simple Mail Transfer Protocol
SNA:	Systems Network Architecture
SNAP:	Shipboard Nontactical ADP Program
SPCC:	Navy Ships Parts Control Center
SS:	System Specification
STE:	Signal Terminating Equipment
SYN:	Synchronize
TAC:	Terminal Access Controller
TCB:	Transmission Control Block
TCP:	Transmission Control Protocol
TELNET:	Network Virtual Terminal Protocol
TP-4:	Transmission Protocol (layer 4)
TSR:	Telecommunication Service Request
TWA:	Trans World Airlines
UADPS:	Uniform Automated Data Processing System
UIC:	Unit Identification Code

URDB:	User Requirements Data Base
VM:	Virtual Machine
WINCS:	WIN Communications Subsystem
WSF:	Weapons System File

LIST OF REFERENCES

1. Navy Fleet Material Support Office, SPLICE Functional Description, FMSO Document No. F94L0-001-9260 FD-SU01B, 14 March 1983

2. Naval Supply Systems Command, Stock Point Logistics Integrated Communications Environment (SPLICE) System Decision Paper III Executive Summary, 1 March 1985

3. Navy Fleet Material Support Office, SPLICE System Specification, FMSO Document No. F94L0-001-9260 SS-SU01C, 16 January 1984

4. Rosner, R. D., Packet Switching, Tomorrow's Communication Today, Lifetime Learning Publications, 1982

5. Department of the Navy Automatic Data Processing Selection Office, Contract N66032-84-D-0002, 17 November 1983

6. Naval Supply System Command, Stock Point Logistics Integrated Communications Environment (SPLICE) Strategic Planning Document, 15 October 1984

7. Tandem Computers Incorporated, Tandem NonStop TXP System, Hardware Architecture, Marketing Description Brochure 109001-0983, 1983

8. Stallings, W., Local Networks, MacMillan, 1984

9. Defense Communications Agency, Defense Data Network, 16 pages, 1985

10. Deputy Secretary of Defense, AUTODIN II Termination, Memorandum, 2 April 1982

11. Defense Communications Agency, Defense Data Network Program Plan, January 1982, Revised May 1982

12. Under Secretary of Defense for Research & Engineering, Defense Data Network (DDN) Implementation, memorandum, 10 March 1983

13. Department of the Navy, OPNAV INSTRUCTION 2070.4, Department of the Navy Policy on use of the Defense Data Network (DDN), 7 March 1984

14. Defense Communications Agency, Defense Data Network Subscriber Interface Guide, July 1983

15. Defense Communications Agency, Defense Data Network Newsletter, November 1984 - April 1985, 15 May 1985

16. Commander, Naval Telecommunications Command, letter, ser 02/2483, 17 October 1983

17. Commander, Naval Telecommunications Command, letter, ser 02/2484, 17 October 1983

18. Commander, Naval Supply Systems Command, Defense Data Network (DDN) Planning Milestones and Cutover Schedule, letter, 0451 A/LM, 0879U, 18 December 1984

19. Commander, Naval Telecommunications Command, Defense Data Network (DDN) Waiver Extension, letter, 2070, ser N5/7639, 5 July 1985

20. Bolt Baranek and Newman, Interface Message Processor - Specifications for the Interconnection of a Host and an IMP, Appendix J, BBN Technical Report 1822, Revised May 1978

21. National Research Council, Executive Summary of the NRC Report on Transport Protocols for Department of Defense Data Networks, RFC939, National Academy Press, February 1985

22. Assistant Secretary of Defense for Communications, Command, Control, and Intelligence, National Research Council Report on Transport Protocols for DoD Data Networks, memorandum to Director, Defense Communications Agency, April 1985

23. Reynolds, J. and Postel, J., Assigned Numbers, RFC 943, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, April 1985

24. Postel, J. B., Simple Mail Transfer Protocol, RFC 821, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, August 1982

25. Russell, D. S., MM User's Manual, MM Version 8140, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, November 1982

26. An MM Primer, Electronic Mail on Context, Center for Information Technology, Stanford University, October 1981

27. Introduction to MM, Electronic Mail Facility on the Context DECSystem-20 at Stanford University, Center for Information Technology, Stanford University, September 1981

28. User's Guide to TOPS-20, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, April 1984

29. Schneidewind, N. F. and Dolk, D. R., A Distributed Operating System Design and Dictionary/Directory for SPLICE, NPS-54-83-015, Naval Postgraduate School, Monterey, CA, November 1983

30. Defense Communications Agency, Defense Data Network (DDN) Synchronous Terminal Support, Defense Data Network Policy Memorandum #85-1, B613, 22 March 1985

31. Postel, J. and Reynolds, J., TELNET Protocol Specification, RFC 854, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, May 1983

32. Postel, J. (editor), Internet Protocol - DARPA Internet Program Protocol Specification, RFC 791, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, September 1981

33. Postel, J., Internet Control Message Protocol - DARPA Internet Program Protocol Specification, RFC 792, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, September 1981

34. Postel, J. (editor), Transmission Control Protocol - DARPA Internet Program Protocol Specification, RFC 793, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, September 1981

35. Postel, J., File Transfer Protocol, RFC 765, Information Sciences Institute, University of Southern California, Marina Del Rey, CA, June 1980

36. Boehm, B. W., "Seven Basic Principles of Software Engineering," The Journal of Systems and Software, 3.3-24 (1983)

37. Parnas, D. L., "On the Criteria To Be Used in Decomposing Systems into Modules," Communications of the Association for Computing Machinery, December 1972

38. Parnas, D. L., "On the Design and Development of Program Families," IEEE Transactions on Software Engineering, Volume SE-2, No. 1, March 1976

39. Bertine, H. U., "Physical Level Protocols," IEEE Transactions on Communications, April 1980

40. Strazisar, V., How to Build a Gateway, IEN 109, Bolt Baranek and Newman, August 1979

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Superintendent Attn: Library Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Computer Technology Programs, Code 37 Naval Postgraduate School Monterey, California 93943-5100	1
4.	Professor Norman F. Schneidewind Code 54Ss Administrative Sciences Department Naval Postgraduate School Monterey, California 93943-5100	1
5.	Commander Naval Supply Systems Command Attn: CDR Dana Fuller, SC, USN (SUP 043) Washington, DC 20376	1
6.	Commanding Officer Navy Fleet Material Support Office Attn: LCDR Ron Nichols, SC, USN (Code 9RL) 5450 Carlisle Pike P. O. Box 2010 Mechanicsburg, Pennsylvania 17055-0787	1
7.	Director Defense Communications Agency 8th & South Courthouse Roads Washington, DC 20305	1
8.	Commanding Officer Navy Management Systems Support Office Naval Air Station Norfolk, Virginia 23511-6694	1
9.	Commander Naval Logistics Command U. S. Pacific Fleet Attn: COMO Phillip F. McNall, SC, USN (Code N41) Pearl Harbor, Hawaii 96860	1
10.	Commanding Officer Navy Finance Center Attn: CAPT Charles E. Baker, SC, USN Anthony J. Celebrezze Federal Building 1240 East Ninth Street Cleveland, Ohio 44199	1
11.	Commander DLA Systems Automation Center P. O. Box P1605 Columbus, Ohio 43216	1

12. Director 1
Defense Logistics Agency
Cameron Station
Alexandria, Virginia 22314

13. Commanding Officer 1
Navy Aviation Supply Office
700 Robbins Avenue
Philadelphia, Pennsylvania 19111

14. Commanding Officer 1
Navy Ships Parts Control Center
P. O. Box 2020
Mechanicsburg, Pennsylvania 17055

15. Commanding Officer 1
Naval Supply Center
Oakland, California 94625

16. CAPT James A. Gillespie, SC, USN (Ret) 1
5604 Admiral Doyle Road
Pensacola, Florida 32506

17. CAPT James T. Rubeck, USN 1
Chief Engineer
USS KITTY HAWK (CV 63)
FPO San Francisco, California 96634-2770

18. LCDR Jonathan B. Schmidt, USN 1
Executive Officer
USS O'CALLAHAN (FF 1051)
FPO San Francisco, California 96674

19. LCDR Stephen M. Carr, SC, USN 2
Navy Management Systems Support Office
Naval Air Station
Norfolk, Virginia 23511-6694

20. Commander 1
Naval Supply Systems Command
Attn: CDR Arden Goss, SC, USN (SUP 0472)
Washington, DC 20376

21. Commander 1
Naval Supply Systems Command
Attn: Ms. Linda Matthews (SUP 0451)
Washington, DC 20376

22. LCDR Ted Case, SC, USN 1
SMC Box 1153
Naval Postgraduate School
Monterey, California 93943

23. Professor Gary K. Poock 1
Code 55PK
Department of Operations Research
Naval Postgraduate School
Monterey, California 93943-5100

24. Mr. Edward M. Carr 1
14733-A Gold Creek Court
Grass Valley, California 95949